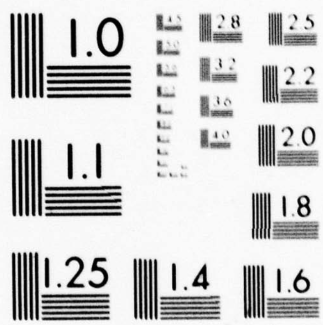
AD-A072 201 NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA F/G 17/1
UNDERWATER SOUND PROPAGATION-LOSS PROGRAM. COMPUTATION BY NORMA--ETC(U)
MAY 79 D F GORDON
UNCLASSIFIED NOSC-TR-393 NL

1 OF 2
AD
A072201


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

LEVEL II

(12)

B.S.

NOSC

NOSC TR 393

AD A072201

DDC
RECEIVED
AUG 1 1979
C

NOSC TR 393

Technical Report 393

UNDERWATER SOUND PROPAGATION-LOSS PROGRAM

Computation by normal modes for layered oceans and sediments

DF Gordon

17 May 1979

Final Report for Period 1976 — 1978

Prepared for
Naval Sea Systems Command
(NSEA 63R-23) Washington DC 20362

NAVAL OCEAN SYSTEMS CENTER
SAN DIEGO, CALIFORNIA 92152

79 08 1 024

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

RR GAVAZZI, CAPT, USN
Commander

HL BLOOD
Technical Director

ADMINISTRATIVE INFORMATION

The computer program described in this report was developed in the course of work sponsored by Naval Sea Systems Command, Sonar Technology Office (NSEA 63R-23), under Problem SF 52-552-602, NOSC work unit 714-SU10. Some elements of this program have been in development since 1965, but the final modifications, to achieve the current capabilities, and the reporting were done from 1976 to 1978. The computer program is derived from an earlier program developed by MA Pedersen. D White did significant parts of the mathematical analysis. This report was approved for publication 17 May 1979.

Released by
MR Akers, Head
Systems Concepts and Analysis
Division

Under authority of
GU Rogers, Head
Ocean Surveillance Systems
Department

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER NOSC Technical Report 393 (TR 393)	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) (6) UNDERWATER SOUND PROPAGATION-LOSS PROGRAM - Computation by normal modes for layered oceans and sediments .	5. TYPE OF REPORT & PERIOD COVERED (9) Final Report, for Period 1976-1978,
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) DF/Gordon (14) NOSC-TR-393	8. CONTRACT OR GRANT NUMBER(s) (16) F52552
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Ocean Systems Center San Diego CA 92152	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NSEA 06H1 (17) SF52-552-602 724-SU10 (12) 104p.
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command (NSEA 63R-23) Washington DC 20362	12. REPORT DATE (11) 17 May 1979
	13. NUMBER OF PAGES 100
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

62759 N

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Ducts	Computer programs
Propagation	Numerical analysis
Acoustics	Continued fractions
Layers	Iterations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A normal-mode program for a sound-speed profile of an arbitrary number of layers has been constructed. It has been used extensively and successfully for 12 years to compute sound propagation in idealized underwater acoustic ducts. Documentation is contained for those who wish to use or modify this program. The FORTRAN statements are given for both the normal-mode and a mode-follower program. The numerical analysis necessary for computing modified Hankel functions of order 1/3 is given. The analysis includes a continued fraction technique.

OBJECTIVE

Translate well-known differential equation solutions into a working program to compute propagation in underwater acoustic ducts. Document the program methods, to assist users of this and similar programs.

RESULTS

1. An effective program for computing propagation loss in a layered ocean by normal modes has been developed. Complete documentation for the program is contained herein.

2. Sediment layers are modeled as fluids in which densities, sound speeds, and absorption can be specified. This permits a complete wave solution for bottom reflected sound energy.

3. A continued fraction technique for evaluating asymptotic series is shown to give superior results in evaluating the auxiliary functions required in this program, the modified Hankel functions of order 1/3.

4. A mode follower program given here is useful in tracing eigenvalues. Such traces are needed to understand the eigenvalue structure.

RECOMMENDATIONS

1. Improve the mode locating ability of this normal-mode program to make it self-contained. It currently requires user interaction to locate eigenvalues.

2. Investigate methods to incorporate the effect of rough boundaries into this program.

CONTENTS

INTRODUCTION

This report describes a normal-mode program that has been used successfully for 12 years to compute sound propagation in idealized underwater acoustic ducts. The theory and considerations used in developing the program are discussed here, and a copy of the FORTRAN statements are included as appendix A. Appendix B consists of sample inputs and outputs to assist users in gaining familiarity with the program. It is hoped that this report contains sufficient information to allow a user to run the program and to modify it as desired.

This program follows the methods developed by Furry and Freehoffer (ref 1) to compute electromagnetic propagation in the 1940s. Marsh adapted these methods to underwater sound in his doctoral thesis (ref 2). Using this material, Pedersen, at NOSC in the late 1950s, adapted the method to digital computers and developed the programs to compute the auxiliary functions. This original program used two layers to define the sound-speed profile (ref 3). This program was expanded to three layers by DF Gordon and RF Hosmer and finally to the multiple-layer program reported here. In this program the only constraints on the number of layers are computer space and running time. The program is normally configured to permit up to 12 layers.

The earlier programs were used to study sound propagation in ocean surface ducts. Programs that permit more layers have proven useful also for studying propagation in the deep ocean, although the number of modes required generally limits computations to frequencies below 300 Hz. The multiple-layer program has also proven useful in modeling sediment layers and thus in computing shallow-water propagation.

The principal limitation in the application of this program to real-world situations is the requirement of ideal conditions: boundaries must be smooth and horizontal, and no variation of boundary conditions with range is permitted. Despite this limitation, the program has proven useful in predicting and explaining acoustic propagation and has applications in a number of related areas. These include checking other types of wave-theory models or corrections such as caustic corrections; determining group velocities, dispersion curves, and reflection coefficients; and determining acoustic coupling between ducts.

The following paragraphs describe the specific topics covered by the sections in this report. In GENERAL SOLUTION are the equations required to solve the wave equation with the boundary conditions used here. DETERMINANT is part of the basic solution but is concerned with the particular numerical method used in this program to evaluate the conditions imposed by the boundaries. Other approaches could be used instead. A later section, NUMERICAL BREAKDOWN, is also part of the basic solution, but deals with special numerical problems that have arisen but are not apparent from the basic equations.

---

1. The Bilinear Modified-Index Profile, by WH Furry, in Propagation of Short Radio Waves, DE Kerr, ed; MIT Rad Lab series, vol 13, p 140–168, McGraw-Hill, New York, 1951.
2. Navy Underwater Sound Laboratory Report 111, Theory of the Anomalous Propagation of Acoustic Waves in the Ocean, by HW Marsh, 1950.
3. Normal-Mode Theory Applied to Short-Range Propagation in an Underwater Acoustic Surface Duct, by MA Pedersen and DF Gordon; J Acoust Soc Am, vol 37, p 105–118, January 1965.

FINDING EIGENVALUES deals with the philosophy of eigenvalue location employed by this program, which essentially leaves this function to the user, the program only serving as a tool. It shows how the program is used to make computations.

Several "automatic" mode finding versions of this program have been developed to the point of accommodating certain classes of profiles. However, they need further development and have not yet been reported.

SOUND SPEED PROFILE indicates the required equations for curve fitting and the various ways the sound speed can be read in on cards. A continuous water profile can be entered quite simply, but sediment layers with sound speed discontinuities and absorption gradients can become complicated.

REFLECTION COEFFICIENTS AND OTHER AUXILIARY OUTPUTS describes a short subroutine that computes reflection coefficients for any mode at a given profile interface. Intermode interference lengths and mode damping coefficients are also discussed.

COMPUTATION OF THE MODIFIED HANKEL FUNCTIONS gives the analysis necessary for computing these functions. The use of continued fractions to evaluate an asymptotic series is discussed. To facilitate running the program on computers of different word length, this section provides the information required to optimize the functions for the different word lengths.

MODE FOLLOWER PROGRAM describes a separate but related program for investigating the eigenvalues themselves rather than using them to compute propagation losses.

GENERAL SOLUTION

The derivation of the normal-mode solution has been discussed from various points of view (eg ref 1, 4, 5). Only an outline is given here. In general, the time-independent wave equation is written in polar coordinates and the azimuthal coordinate is dropped under the assumption that the field is independent of azimuthal direction. Thus

$$(1/r)\,(\partial/\partial r)\,[r(\partial\psi/\partial r)] + (\partial^2\psi/\partial z^2) + (\omega^2/c^2)\,\psi = 0, \qquad (1)$$

where ψ is the velocity potential, c the sound speed, and the independent variables are depth, z, and range, r.

Equation (1) is then separated into range- and depth-dependent parts with a separation constant λ. The separation is possible when the sound speed is a function of depth only. After accounting for the source discontinuity and the outgoing radiation condition, integrating over all real values of the separation constant, and normalizing, one can find the solution for a field point in terms of propagation loss H as follows:

---

4. Naval Air Development Center Report NADC-72002-AE, Normal Mode Solutions and Computer Programs for Underwater Sound Propagation, by CL Bartberger and LL Ackler, 4 April 1973.
5. A Normal Mode Theory of an Underwater Acoustic Duct by Means of Green's Functions, by RL Deavenport; Radio Sci, vol 1, p 709-724, 1966.

$$H = -10 \log \left| \rho_s \rho_h \pi \sum_{n=1}^{N} H_0^2 (\lambda_n r)\, U_n(z)\, U_n(z_0) \right|^2 + \alpha_A r, \qquad (2)$$

where r is the range, z_0 is the source depth, z is the receiver depth, H_0^2 is the Hankel function of order zero, second type, λ_n is the *nth* eigenvalue, U_n is the depth function for mode n, and ρ_s and ρ_h are the densities at source and receiver. The sum is over the number of modes, N, making a significant contribution. The final term contains the volume attenuation coefficient, α_A. From Thorp (ref 6), α_A in dB/m is computed by the relationship

$$0.9144\, \alpha_A = 0.0001\, F^2/(1 + F^2) + 0.04\, F^2/(4100 + F^2), \qquad (3)$$

where F is the frequency in kHz. Improved equations or those for specific ocean areas can be easily substituted. The depth function, U_n, is a solution to the depth-dependent part of the separated wave equation

$$d^2U/dz^2 + [\omega^2/c^2(z) - \lambda^2]\, U = 0, \qquad (4)$$

where

$$\omega = 2\pi f$$

and f is the frequency, in Hz.

A closed-form solution to eq (4) can be obtained when the reciprocal sound speed squared or squared index of refraction is a linear function of depth. That form is used in this program, and sound speed in each layer is expressed as follows:

$$[c_i/c(z)]^2 = 1 - 2\, \gamma_i\, (z - z_i)/c_i, \qquad (5)$$

where c_i, z_i, and γ_i are the sound speed, depth, and sound-speed gradient, respectively, at the top of layer i. Up to 12 such layers are permitted by the program, for modeling the sound-speed profile.

With this expression for sound speed, solutions to eq (4) can be expressed in terms of solutions to Stokes' equation

$$h'' + zh = 0. \qquad (6)$$

Only a simple change in independent variable is required from z to ζ, where

$$\zeta_i(z) = \left[a_i^3 (z - z_i) + \omega^2/c_i^2 - \lambda^2 \right] / a_i^2 \qquad (7)$$

and

$$a_i^3 = -2\gamma_i\, \omega^2/c_i^3. \qquad (8)$$

6. Analytic Description of the Low-Frequency Attenuation Coefficient, by WH Thorp; J Acoust Soc Am, vol 42, p 270, 1967.

The solutions to Stokes' equation that are used are the modified Hankel functions of order 1/3, $h_1(\zeta)$ and $h_2(\zeta)$. The depth function is a linear combination of these two independent solutions:

$$F_{n,i}(z) = A_{n,i}h_1(\zeta_n) + B_{n,i}h_2(\zeta_n), \tag{9}$$

where F_n is the unnormalized form of U_n. The coefficients $a_{n,i}$ and $B_{n,i}$ for mode n in layer i are determined to satisfy boundary conditions, which will be listed below. Values of λ_n for which the boundary conditions can be satisfied are the eigenvalues.

The first boundary condition is the radiation condition. It is satisfied by using a negative sound-speed gradient in the deepest layer, which extends to infinite depth, and by letting the depth function there be proportional to h_2 only. That is,

$$F_n(z) = B_n h_2(\zeta_n). \tag{10}$$

At the surface the depth function is zero:

$$F_n(0) = 0, \tag{11}$$

and at layer interfaces, ρU and its depth derivative are continuous:

$$\rho_i F_{n,i}(z) = \rho_{i+1} F_{n,i+1}(z); \tag{12}$$

$$dF_{n,i}(z)/dz = dF_{n,i+1}(z)/dz. \tag{13}$$

Here ρ_i is the density in layer i, and the excess acoustic pressure, p, is given by

$$p = \rho U.$$

If U is assumed to be the vertical component of the velocity potential, eq (12) and (13) are equivalent to requiring that the pressure and the vertical component of particle velocity be continuous across the layer interface.

Applying these boundary conditions to a sound-speed profile consisting of M layers results in 2M − 1 linear equations in h_1 and h_2. They are homogeneous in that the constant is zero in each equation. There are M−1 coefficients A_i to be determined and M coefficients B_i. These coefficients can therefore be determined within a constant of proportionality D, provided the system of equations is linearly dependent. That is, the 2M − 1 square matrix of coefficients of A_i and B_i must be of rank 2M − 2 or less. Its determinant will then be zero. This is the eigenvalue condition. Values of λ must be found which make the determinant zero. This determinant, G, is discussed in more detail in a later section.

Zeroes of the determinant, G, are found by using the secant method. The variable in this iterative method can as well be some function of λ as λ itself, and we use the following complex phase velocity (v):

$$\lambda_n = \omega/v_n. \tag{14}$$

To find a v that is a root of G requires an initial guess, v_1, where the subscript 1 refers to the step in the iteration and a small increment, δ_1. Each succeeding estimate is given by the relationship

$$v_{j+1} = v_j + \delta_j,$$

where

$$\delta_j = -(v_j - v_{j-1})\, G_j/(G_j - G_{j-1}). \tag{15}$$

The details of this iterative process are given in a later section.

When an eigenvalue v_n is found, the coefficients are then evaluated. One coefficient can be given an arbitrary value, so A_1 is set to $\rho_1 h_2[\zeta_1(0)]$. From eq (11), B_1 is then $-\rho_1 h_1[\zeta_1(0)]$. Pairs of equations (eg (12) and (13)) for each successive interface can then be used to evaluate the next A_i and B_i as discussed later.

Finally the normalizing factor, D_n, for mode n is obtained by the relationship

$$D_n = \int_0^\infty \rho F_n^2(\zeta)\, dz. \tag{16}$$

This equation follows from the orthogonality of the depth functions. It is not the pressure, however, which is proportional to ρU, but $\rho^{1/2}U$ that is orthogonal (ref 7). Therefore, D_n must be determined such that the integral of ρU^2 is 1.

From Stokes' equation (eq (6)) and eq (7-9), the integral of F^2 takes the form

$$\int_{z_i}^{z_{i+1}} F^2(\zeta)\, dz = \left[\zeta_i(z)\, F^2(\zeta)\Big/a_i + F'^2(\zeta)\Big/a_i^3 \right]\Bigg|_{z_i}^{z_{i+1}}. \tag{17}$$

Therefore

$$D_n = -\rho_1^3 W^2/a_1 + \sum_{i=1}^{n-1} \left\{ \rho_i[\zeta_i(z_{i+1})/a_i - \rho_i \zeta_{i+1}(z_{i+1})/(a_{i+1}\rho_{i+1})]\, F_i^2(z_{i+1}) \right.$$

$$\left. + \left(\rho_i/a_i^3 - \rho_{i+1}/a_{i+1}^3\right) F_i'^2(z_{i+1}) \right\}, \tag{18}$$

where eq (12) and (13) have been used to combine terms at each interface. The derivative of F takes the form

$$F_i'(z_{i+1}) = a_i \left\{ A_i h_1'[\zeta_i(z_{i+1})] + B_i h_2'[\zeta_i(z_{i+1})] \right\}. \tag{19}$$

The Wronskian, W, is an imaginary constant (see eq (85)) and is the contribution of eq (17) at the surface:

$$W = -1.45749544104i.$$

---

7. Some Effects of Velocity Structure on Low-Frequency Propagation in Shallow Water, by AO Williams; J Acoust Soc Am, vol 32, p 363-365, March 1960.

The depth functions are normalized by the relationship

$$U_n(z_0)\, U_n(z) = F_n(z_0)\, F_n(z)/D_n. \tag{20}$$

The functions F and F′ used in computing D_n are conveniently assembled from the elements of the determinant and the coefficients A_i and B_i. This requires care in developing the computer code, because F is always multiplied by ρ and F′ has the term a_i in it. The surface differs from the other layers in that F_1 is zero there and F'_1, by eq (19), is $a_1 W$. However, because ρ_1 appears as a factor in the coefficients of F_1, the actual value of F'_1 at the surface in the computation is $\rho_1\, a_1 W$. This factor of ρ_1 together with the $\rho_1^{1/2}$ needed for orthogonality, when squared, gives the ρ_1^3 of eq (18).

DETERMINANT

Normal modes are determined by finding the eigenvalues of a characteristic equation which, in turn, is obtained by setting a determinant to zero. The determinant is obtained from the coefficient matrix of a set of linear, homogeneous equations expressing the boundary conditions as given by eq (10) – (13). Since the method of handling this determinant is a central feature of this normal-mode program, it is given in detail here.

The first line of the matrix is taken from eq (11) as

$$B_1\, \rho_1\, h_2\, [\zeta_1(0)] + A_1\, \rho_1\, h_1\, [\zeta_1\,(0)] = 0. \tag{21}$$

At each profile interface, i, where i numbers the interfaces below the surface from 1 to N–1, the two boundary conditions given by eq (12) and (13) are

$$B_i\, \rho_i\, h_2\, [\zeta_i\,(z_{i+1})] + A_i\, \rho_i\, h_1\, [\zeta_i\,(z_{i+1})] - B_{i+1}\, \rho_{i+1}\, h_2\, [\zeta_{i+1}\,(z_{i+1})]$$
$$- A_{i+1}\, \rho_{i+1}\, h_1\, [\zeta_{i+1}\,(z_{i+1})] = 0 \tag{22}$$

and

$$B_i\, a_i\, h'_2\, [\zeta_i\,(z_{i+1})] + A_i\, a_i\, h'_1\, [\zeta_i(z_{i+1})] - B_{i+1}\, a_{i+1}\, h'_2\, [\zeta_{i+1}\,(z_{i+1})]$$
$$- A_{i+1}\, a_{i+1}\, h'_1\, [\zeta_{i+1}\,(z_{i+1})] = 0\,. \tag{23}$$

The coefficients of A_i in the first equation and B_{i+1} in the second will be the diagonal elements of the matrix. The nonzero elements of the matrix will therefore be no more than two places from the diagonal. The matrix can be stored in the computer in an array of size (2M–1) X 4, where M is the maximum number of layers in the sound-speed profile. In the final layer, $A_N\, h_1$ is omitted, as in eq (10). In the program, the real and imaginary parts are stored in separate arrays.

The sparseness of the matrix permits efficient evaluation by a triangularization process of row reduction. For each pair of rows representing a pair of equations given by eq (22) and (23), the first element from the first equation and the first two from the second equation must be set to zero by subtracting the proper multiple of preceding rows. The determinant is then the product of the diagonal elements of the triangularized matrix. The value of the determinant, G, is used in eq (15) to find the roots by iteration.

Note that a value of v that makes this determinant zero, or near zero, ordinarily is zero because only one diagonal element is very small. For trapped modes this element is at the row representing the first interface below the mode, ie the interface just below the layer of positive gradient in which the sound speed is equal to the mode phase velocity. For unstrapped modes it is usually the final diagonal element that is small. Thus the layers in which the sound speed is greater than the phase velocity of a mode do not greatly affect the eigenvalue. Eigenvalues are determined mainly by those parts of the sound-speed profile that are less than the phase velocity.

When an eigenvalue is found, the coefficients A_i and B_i must next be evaluated. As mentioned earlier, one coefficient can be arbitrarily chosen. This is done, and eq (21) is satisfied by letting

$$A_1 = \rho_1 h_2[\zeta_1(0)]$$

and

$$B_1 = -\rho_1 h_1[\zeta_1(0)]. \tag{24}$$

The factor ρ_1 is used simply because the number containing it is easily available in the program. It is divided out by the normalizing factor, D. Eq (22) and (23) can then be used to evaluate the remaining coefficients, but the triangularized form of the matrix yields the coefficients with less computation. If g_{ij} is the element in the *ith* row and *jth* column of the triangularized matrix, then by Cramer's rule,

$$B_i = A_{i-1}\, g_{2i-2,2i-2}\, g_{2i-1,2i}/E_i$$

and

$$A_i = -A_{i-1}\, g_{2i-2,2i-2}\, g_{2i-1,2i-1}/E_i,$$

where

$$E_i = g_{2i-2,2i-1}\, g_{2i-1,2i} - g_{2i-2,2i}\, g_{2i-1,2i-1}. \tag{25}$$

A simpler form is used for B_N in the final layer since there is no A_N there.

In certain situations numerical problems can arise in evaluating the determinant. These require some extra tests in the subroutine that makes the evaluation. The extra tests will be discussed in the section, NUMERICAL BREAKDOWN. A more routine problem is the loss of accuracy that can arise in subtractions in the row reduction of the matrix. This loss results in less sharpness of convergence to a root. The size of the determinant, G, can be 14 orders of magnitude less at a root than at the general background near the root. This variation occurs because the modified Hankel functions can be computed to about 14-place accuracy in a computer with 18 decimal places available. Modes usually converge to 10 or 12 places; thus a few places are lost in evaluating the determinant. In some profiles, usually those with multiple ducts or those in which propagation through bottom sediments plays a large part, the convergence can be much poorer. Modes need to converge to about 4 places to be reliable for computing losses, and convergence occasionally fails to meet this requirement. The only current cure for this loss in accuracy is to go to higher-precision arithmetic or to compute the modified Hankel functions to greater accuracy. For instance, a standard

matrix triangularization routine that uses full row and column pivoting has been tried with no resultant increase in accuracy.

FINDING EIGENVALUES

There are versions of this program under development that will locate the eigenvalues and do the entire computation without user intervention. Currently, however, these versions are reliable only for the simpler types of profiles – usually those with only one duct – and are not ready to be reported. Locating eigenvalues with the standard version of the program is discussed here.

The standard version of the program requires the user to find the eigenvalues. In this version, each time an eigenvalue is determined by iteration, the resulting value is stored and counted as an eigenvalue. Therefore, the user must ensure that all iterations result in good roots, that all required modes have been determined, and that no modes are present more than once. In most cases the user must expect to make more than one computer run to obtain this result.

CONTROL CARDS

The user controls the eigenvalue determination by using any of four different types of control cards. The first type specifies an initial value for v and an initial step size, Δv. These are both complex numbers with a real and an imaginary part. G is then evaluated at v and at $v + \Delta v$ to start the iteration. These are essentially the v_j and v_{j+1} of eq (15). If these two trial eigenvalues are in the vicinity of a root, the iteration will converge to that root.

The second type of card specifies a line segment in the complex plane, along which a search for eigenvalues, v, is made. The end points of the line are given along with the number of equally spaced points at which the line is to be divided. G is then evaluated at each successive division point along the line until a relative minimum in $|G^2|$ is found, indicating that a root is nearby. The iterative process is applied to find the root. The initial step size, Δv, is first computed to bring the second evaluation at $v + \Delta v$ as close as possible to the true root. This is done by using the point which resulted in minimum $|G^2|$ and the points on either side of it to determine the minimum of the parabola passing through them. If $v - h$, v, and $v + h$ are the three points at which G was evaluated, it follows that the distance from v to the minimum of the parabola

$$\Delta v = h[G(v + h) - G(v - h)]/2[2G(v) - G(v + h) - G(v - h)]. \qquad (26)$$

When the iteration is complete, the eigenvalue is recorded and the program continues to step along in the direction of the given line, checking again for a minimum. However, the stepping is resumed from the newly located root rather than from the approximate location where the minimum was detected. With this correction in position, the designated line does not have to hug the curve on which the eigenvalues are located because it is corrected at each eigenvalue.

This method of finding eigenvalues has proven very successful. Its main utility arises, though, because the eigenvalues of the trapped modes have negligible imaginary parts and the

search can be made along the real line. In simple profiles this can often give a successful set of modes on the first try. Usually, only the three initial eigenvalues need to be located by this means because further eigenvalues can be located by extrapolation on the previous three. This is the function of the third type of control card.

The third type of card specifies the number of additional modes to be determined by extrapolation. The starting value of each eigenvalue is determined by extrapolating from the three most recently determined eigenvalues to find v. The step size, Δv, is chosen as 0.0001 times the distance between the last two eigenvalues. The exact eigenvalue is then determined by iteration. The extrapolation is the simple parabolic form for equal steps:

$$v = 3v_n - 3v_{n-1} + v_{n-2}. \tag{27}$$

This method of locating modes works well when the modes lie along a smooth curve, as usually occurs for single ducts. But this relationship does not always occur for profiles with multiple ducts.

The final control card is punched by the program when requested and contains the correct eigenvalue to full precision. Upon encountering this card, the program does not iterate, but instead evaluates G for this eigenvalue and stores this value of G as the next eigenvalue. A deck of such completed eigenvalues can be stored, saving the expense of recomputing the eigenvalues for a given profile and frequency.

ITERATION TERMINATION

A full description of the iteration of eq (15) should include the method of termination. The usual criterion for stopping is that G fails to become smaller. As G approaches minimum size, however, round-off error can act as noise so that G is no longer a predictable function of v. The denominator of eq (15) can then be very small by chance, resulting in a large value for δ_i. If this happens, the next value of v, which was as near to the root as possible, will be far away. A much better convergence criterion is that δ_i has reached a minimum in absolute value. In the program, iteration is stopped when $|\delta^2|$ exceeds the previous value by a factor of 2. However, this criterion is not applied until three iterative steps have been completed, to permit the process to become well established. An upper limit of 15 iterative steps is permitted. We have not found an improvement on the root after 15 steps.

SOUND SPEED PROFILE

The normal mode program requires as inputs the depth of each layer and the sound speed and sound speed gradient at the top of each layer. These variables are mapped into the dimensionless internal variables of the program by eq (7). The purpose of the sound speed profile processing portion of the program is to accept the profile parameters in a form convenient for the user and to translate them into the required sound speeds and gradients.

The first function of the processing program is to make the sound speed continuous at interfaces. This is done simply by using the sound speed at the bottom of one layer as the sound speed at the top of the next. It may be necessary to compute the sound speed at the bottom of the layer. The necessary parameters will have been given. Occasionally a

discontinuity in sound speed is required, as when modeling an interface between water and sediment. The user indicates this by specifying the sound speed at the top of the layer. If left blank, the program provides the sound speed necessary for continuity.

A second function of the processing program is to permit a layer to be defined by the sound speed at top and bottom of the layer rather than by one sound speed and one gradient. Note that the profile form as given by eq (5) is a two-parameter curve.

The last layer extends to infinite depth, so a gradient must be specified at the top of it. However, this gradient can be specified by giving a depth and sound speed point below the last layer. The program handles this by checking to see if the gradient of the last given layer is unspecified. If it is, the number of layers is reduced by one, which causes the last layer to be only the required extra point determining the final gradient. This final gradient must always be negative, as is required by the boundary conditions. The program user must ensure that this gradient is negative and that no gradient is zero. A zero gradient will appear in the denominator of eq (7).

These functions of the profile processing program are relatively simple, but an additional capability used to model sediment bottoms greatly increases the complexity of the program. The capability required is to specify the absorption in a layer by adding an imaginary part to the sound speed. In older versions of this normal mode program an imaginary part, expressed as an absorption coefficient, could be added to the sound speed at the top of the layer. This imaginary part is small compared to the real part. Since the gradient was assumed real at the top of the layer, the imaginary part was initially not changing with depth and it usually changed only a minor amount through the depth of the layer. However, this small change could not always be relied upon. Also Hamilton (ref 8) has published data on absorption gradients in sediment layers, so more precise control of this part of the sound speed function is needed to model sediment layers. Therefore, a more comprehensive profile processing routine has been incorporated in the normal mode program. This curve-fitting process is described below.

The following quantities can be input for each layer depth starting at the surface:

Depth of top of the layer

Sound speed at top of layer

Sound speed at bottom of layer

Real part of sound speed gradient at top of layer

Attenuation in loss per km at the top of the layer

A similar attenuation at the bottom of the layer

Density in the layer

The density is a constant in the layer and as such requires no further curve fitting. Redundant parameters are left blank on input cards. In some cases negative values serve as flags to indicate specific treatment. For instance a negative value of absorption at the top of a layer

8. Sound Attenuation as a Function of Depth in the Sea Floor, by EL Hamilton; J Acoust Soc Am, vol 59, p 528-535, March 1976.

directs the program to use the same imaginary part of sound speed as occurred at the bottom of the previous layer. Similar flags at the bottom of a layer are discussed later.

Absorption per Hz is given in units of decibels per km (or kiloyard). The quotient of absorption over frequency is used because Hamilton (ref 8) usually considers absorption (or attenuation) as proportional to frequency with a coefficient k. We use the symbol h instead. That is,

$$\alpha = hf.$$

We interpret α to be in units of dB per km and f in Hz, whereas Hamilton uses dB per m and kHz; but the coefficients h and k remain equal.

The complex wave number in layer i is represented as

$$k_i = \omega/C_i$$

$$= \omega \operatorname{Re}C|C|^{-2} - i\omega \operatorname{Im}C|C|^{-2}. \tag{28}$$

A plane wave will be attenuated α dB per km if

$$\operatorname{Im}k_i = -\alpha/(20\,000 \log e)$$

$$= -\pi A f, \tag{29}$$

where

$$A = h/(20\,000\,\pi \log e).$$

By equating the imaginary part of k_i in eq (28) and (29), the imaginary part of C_i is found to be as follows:

$$\operatorname{Im}C_i = 1/A - [1/A^2 - (\operatorname{Re}C_i)^2]^{1/2}. \tag{30}$$

If α is zero, which is the case usually used in water layers, eq (30) cannot be used; but the imaginary part of C is then simply zero. These two cases are treated separately in the program.

When sound speed is given at the top and bottom of layer i, the imaginary parts of the sound speeds are determined by eq (30) and the only curve fitting task is to determine the gradient γ_i. Solving eq (5) for γ_i,

$$\gamma_i = C_i(C_{i+1}^2 - C_i^2)/2C_{i+1}^2(z_{i+1} - z_i). \tag{31}$$

The gradient is a complex number since the C's here are complex. The z's are real.

A second version of this computation arises if the gradient is required to be a real number. In this case, which is used to match older versions of the program, an additional parameter must be left unspecified and this parameter is Im C_{i+1}. This is equivalent to having the sound absorption at the bottom of the layer unspecified. Therefore, a negative number input for this parameter is used as a flag to call for this particular fitting procedure.

For this situation, given Re C_i, Im C_i, Re C_{i+1}, and making γ_i real, the determination of γ_i and Im C_{i+1} is not simple. When γ_i is eliminated from the real and imaginary parts of eq (31), a quartic equation in Im C_{i+1} results. Rather than derive an algebraic solution to this equation, it is solved by iteration under Newton's method. A good first guess at the solution is Im $C_{i+1} \cong$ Im C_i. Four iterations usually give an accurate root. The equation is

$$\begin{aligned} &\mathrm{Im}\, C_i (\mathrm{Im}\, C_{i+1})^4 + \left[\mathrm{Im}(C_i)^3 + 2(\mathrm{Re}\, C_{i+1})^2\, \mathrm{Im}\, C_i\right] (\mathrm{Im}\, C_{i+1})^2 \\ &\qquad + 2\mathrm{Re}\, C_{i+1}\, \mathrm{Re}(C_i)^3\, \mathrm{Im}\, C_{i+1} \\ &\qquad + \mathrm{Im}\, C_i (\mathrm{Re} C_{i+1})^4 - (\mathrm{Re}\, C_{i+1})^2\, \mathrm{Im}(C_i^3) = f(\mathrm{Im}\, C_{i+1}). \end{aligned} \tag{32}$$

The root is then found:

$$(\mathrm{Im}\, C_{i+1})_j = (\mathrm{Im}\, C_{i+1})_{j-1} - f/f'.$$

The gradient, γ, is next given by the relationship

$$\gamma_i = \left\{ \mathrm{Im}\, C_i \left[(\mathrm{Re}\, C_{i+1})^2 - (\mathrm{Im}\, C_{i+1})^2\right] + 2\mathrm{Re}\, C_i\, \mathrm{Re}\, C_{i+1}\, \mathrm{Im}\, C_{i+1} - \mathrm{Im}(C_i^3)\right\} \Big/ \\ [4\, \mathrm{Re}\, C_{i+1}\, \mathrm{Im}\, C_{i+1}\, (z_{i+1} - z_i)]. \tag{33}$$

Because the root of eq (32) may not be exact, Im γ_i may not be exactly zero. This slight error can be transferred to C_{i+1} by using the computed real γ_i to recompute C_{i+1}. This is done in the program by transferring to a portion of the program already designed to do this.

When sound speed and gradient at the top of the layer are given, the parameters required by the program are all given. The sound speed at the bottom of the layer is routinely computed, however, because it may be required to make the next layer continuous. Equation (5) is used to determine the sound speed at depth z_{i+1}, which is the depth of the bottom of the layer. This is straightforward, but several complications arise. Only the real part of the gradient at the top of the layer is used as an input because situations have not arisen that require that the imaginary part of the gradient be specified. Often the attenuation is given at both top and bottom of the layer. That is, Re C_i, Im C_i and Re γ_i are given, plus a relationship between Re C_{i+1} and Im C_{i+1}. The imaginary part of the gradient, Im γ_i, must be determined as well as both real and imaginary parts of the sound speed at the layer bottom. The derivation of this case is not trivial.

One relationship between the real and imaginary parts of the sound speed is given by eq (28) and (29). From these equations at C_{i+1} we derive

$$A(T - i) = 2/C_{i+1}, \tag{34}$$

where

$$T = \mathrm{Re}\, C_{i+1} / \mathrm{Im}\, C_{i+1}.$$

Substituting this expression for C_{i+1} into eq (31) and equating real parts gives a quadratic expression for T which has a usable root of

$$\mathrm{Re}(C_i^3)T = -\mathrm{Im}(C_i^3) - \left\{ [\mathrm{Im}(C_i^3)]^2 + \mathrm{Re}(C_i^3)B \right\}^{1/2}, \tag{35}$$

where

$$B = \mathrm{Re}(C_i^3) - 8 \,\mathrm{Re}\, \gamma_i(z_{i+1} - z_i)/A^2 + 4 \,\mathrm{Re}\, C_i/A^2.$$

From eq (34),

$$\mathrm{Re}\, C_{i+1} = 2T/A(T^2 + 1)$$

and (36)

$$\mathrm{Im}\, C_{i+1} = RC_{i+1}/T.$$

The gradient can now be evaluated by eq (31) to find its imaginary part.

Equations (34) and (35) cannot be used if the attenuation at the bottom of the layer is given as zero. Therefore an alternate form must be used. This form is much simpler than the previous case, since C_{i+1} is real.

$$C_{i+1} = \left\{ \mathrm{Re}(C_i^3) \Big/ [\mathrm{Re}\, C_i - 2 \,\mathrm{Re}\, \gamma_i\, (z_{i+1} - z_i)] \right\}^{1/2} \tag{37}$$

$$\mathrm{Im}\, \gamma_i = [\mathrm{Im}\, C_i - \mathrm{Im}(C_i^3)/C_{i+1}^2]\, [2(z_{i+1} - z_i)]^{-1} \tag{38}$$

Finally, if the special case, γ_i real, is specified by inputting a negative value for absorption, eq (31) can be used directly to give

$$C_{i+1}^2 = C_i^3/[C_i - 2\gamma_i(z_{i+1} - z_i)]. \tag{39}$$

To evaluate the square root, let

$$C_{i+1}^2 = a + bi.$$

Then

$$\mathrm{Re}\, C_{i+1} = \left\{ \left[a + (a^2 + b^2)^{1/2} \right] \Big/ 2 \right\}^{1/2} \tag{40}$$

and

$$\mathrm{Im}\, C_{i+1} = b/2 \,\mathrm{Re}\, C_{i+1}. \tag{41}$$

NUMERICAL BREAKDOWN

A situation arises frequently in which a very small depth function must be computed from the difference of two large numbers. A wrong answer results if this accuracy loss exceeds the word size of the computer. The best way that has been found to avoid this is to check for it within the program and arbitrarily replace the wrong number. In checking for this, a constant, called T-lim in the computer program, is compared to the argument of th

modified Hankel functions or to the argument of the exponential function within modified Hankel functions. A T-lim value of 25.0 is used in the program, but a smaller number occasionally is required. The program user can alter T-lim by appropriate input cards (Key 8 = 1 followed by a new value of T-lim). The next few paragraphs demonstrate the symptoms of this problem, so as to assist a user in recognizing the problem. The remainder of this section describes the modifications that have been made to the computer program to correct this loss of accuracy.

The solid line of figure 1 shows a simple surface duct and the phase velocities of the first three modes at 3 kHz. For this profile, the depth function of mode 1 is shown in figure 2. The solid line is the depth function as computed by a program that does not correct for numerical breakdown. The dashed line shows the correct depth function below a depth of 71 m. This result was determined from Airy functions, not from the program. Between depths of 71 to 100 m, the program *cannot compute the depth function* accurately. In the second layer, which starts at a depth of 100 m, the function can be computed accurately but it is incorrectly placed by the boundary condition that requires the depth functions to be continuous at interfaces. The slope of the depth function was correctly computed as indicated by the identical shape of the three depth functions in the second layer. The shape is such as to make the correct depth function continuous in slope across the interface.

The breakdown in accuracy at a depth of 71 m occurred when ζ had a value of -8.4. (ζ is given by eq (7) and is the argument of the modified Hankel functions.) A negative value of ζ occurs when the mode phase velocity is less than the speed of sound. Since the ray of the same phase velocity cannot reach such a region, the sound field *there is a diffracted field*. The mode depth function is therefore small at such depths. In the figure, the depth function amplitude at the breakdown point is about 7 orders of magnitude (or in terms of propagation loss, 140 dB) down from its maximum. Equations (62), (66), and (68), which will be given for the modified Hankel functions, indicate that the argument of the exponential term is $2/3(8.4)^{3/2}$, or 16.2. The functions h_1 and h_2 will thus be about 10^7 in magnitude at a depth of 71 m. These large values and their small difference account for the approximate accuracy loss of 14 decimal places, which is the general accuracy of the modified Hankel functions.

Incorrect behavior in the depth function usually occurs when ζ is about -8.4. In some more complicated profiles, however, where accuracy is lost in row reduction of the determinant, the depth functions may become incorrect at values of ζ that are less in absolute value. When this problem occurs it can be diagnosed by plotting the depth function of the mode and noting the steep positive slope through some depth interval as in figure 2. When that occurs, the value of T-lim should be decreased.

Incorrect depth functions can cause errors in propagation loss computations in two ways. In figure 2, the solid-line depth function, because of its large size, can cause losses to be too low at a depth of around 100 m. The second error would occur if the duct were deeper, say 110 m. At this depth the erroneous segment of depth function in figure 2 would reach a value of about 10^{-1}, where it would be larger than the correct lobe of the depth function near the surface. With this extra area under the curve, the normalizing factor would be increased significantly and would reduce the size of this entire depth function. Thus, losses near the surface would be larger because of the loss in size of mode 1.

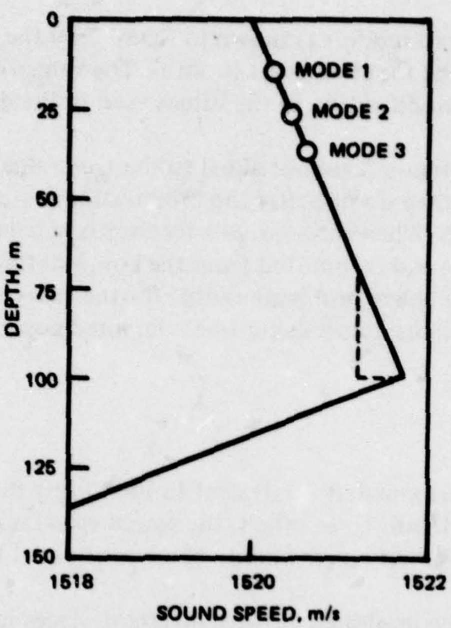


Figure 1. A two-layer sound speed profile for a surface duct. The phase velocities of the first three modes at 3 kHz are marked. The broken line shows a modification of the upper layer to prevent numerical breakdown in mode 1.

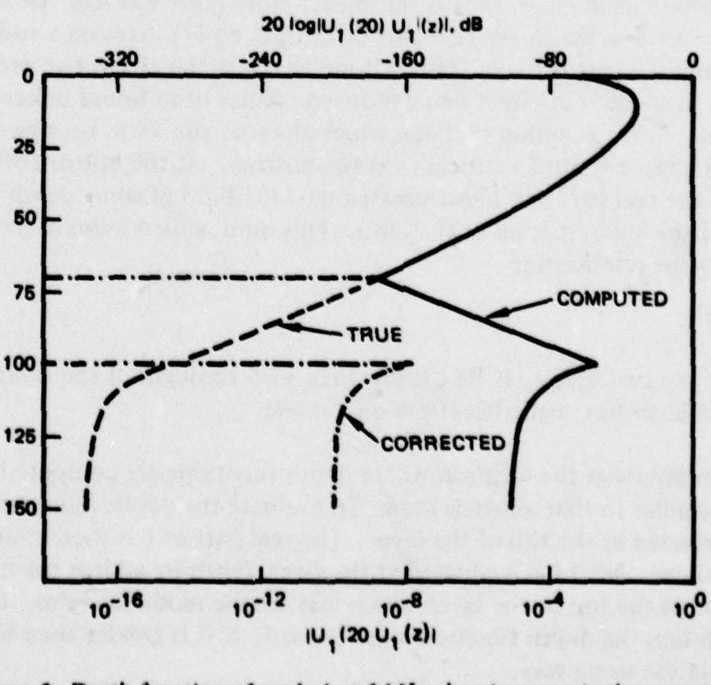


Figure 2. Depth function of mode 1 at 3 kHz, showing error in the computed function. The true function cannot be computed without increasing computer word length, but the corrected value can and it will not cause a large error in the mode sum.

The standard correction to mode 1 is shown in figure 2 by the dot-dashed line. In the depth interval where $\zeta < -8.4$, the function is set to zero. The values of the depth function at greater depths result from a modification in the values used in the determinant.

The corrected values in figure 2 are not equal to the true value of the depth function, but they are small enough that they do not alter the propagation loss to a tenth of a decibel when a full set of modes is used. When the source or receiver is at a depth where such corrections are necessary, the mode can be omitted from the computation. Thus, properly omitting modes would solve the above problems except for the cases where the normalizing factor, D, is affected. In these cases, losses cannot be computed accurately without the corrections.

PROGRAM MODIFICATIONS

The modification is approximately equivalent to modifying the sound speed profile as shown by the broken line in figure 1. In effect, the sound speed is not allowed to become enough greater than the phase velocity of the mode being considered to cause problems.

The limitation on ζ is accomplished at three different places in the normal mode program. It is not clear that this is the best way to handle the problem and it may be redundant, but it appears to be an adequate solution. These three corrections will be described next. Finally a correction to the determinant program is described which is necessary because the limiting of ζ can cause false zeroes in the determinant.

In the subroutine SETUP the elements of the determinant are computed by determining ζ at the top and bottom of each layer and then calling the modified Hankel function program. At the top of each layer, Re ζ is set to -7.5 if its value was less. However, this is done in an iterative loop in which the real part of ω/C_i in eq (7) takes on a sufficiently larger value while its imaginary part is fixed. This is done to retain the absorptive properties of a layer when its sound speed is in effect being reduced. It has been found unnecessary to make the above constant, -7.5, a function of T-lim which the user can vary, because an oversized value at the top of a layer is not as critical as at the bottom. At the bottom of a layer, several tests are made. If the real part of ζ has decreased past the limit at some depth between the top and bottom of the layer, it is set at the limit. This limit, called S-lim in the program, is related to T-lim by the relationship

$$S = -(T)^{2/3} \qquad (42)$$

where S and T are the two limits. If Re ζ is less than -7.5 throughout the layer, it is simply set at -7.5. Such a layer has negligible effect on a mode.

In program MAIN at the location where depth functions are computed for given depths, a process similar to that above is used. To evaluate the depth function in a given layer, ζ is first evaluated at the top of the layer. The real part of ζ is then limited as in the program SETUP above. Next ζ is evaluated at the given depth by adding the depth-dependent part onto the value at the top of the layer which may be the modified value. If this final value is less than S-lim, the depth function is set to zero. If it is greater than S-lim, the function is computed in the usual way.

The imaginary part of ζ can be large if the eigenvalue has a large imaginary part or if the speed of sound in the layer has a large imaginary part. When this happens the imaginary part of $2/3\ \zeta^{3/2}$, which appears as an exponential in the modified Hankel functions, may become large in absolute value even though Re ζ has been limited. A final check is therefore made before the exponential is computed. If Im $\zeta^{3/2}$ is greater than T-lim, ζ is reduced in amplitude to the size at which it will equal T-lim. The angle of ζ in the complex plane is preserved.

This limitation of the exponential can be viewed in another way. In a following section the two components of the modified Hankel functions, F_1 and F_2, eq (68) and (69), have exponential terms whose arguments are equal and opposite in sign. When these arguments have magnitude of 2/3 T-lim, they differ in size by 15 decimal places, which is near the 18-decimal-place word size of the machine. The ability to compute the difference in these two terms is essentially the same as the ability to compute the depth function accurately.

PREVENTING ZEROES IN THE DETERMINANT

Placing limits on ζ can cause problems in the determinant because ζ may be set equal to S-lim at several interfaces. The equations that arise for matching boundary conditions may then be identical for these interfaces and may therefore fail to be linearly independent. The triangularized determinant will thus have zeroes on the diagonal at positions equivalent to interfaces that do not have real physical importance for the mode. These will prevent location of the significant "zeroes" or roots. These artificial zeroes must be removed.

The artificial zeroes are detected and removed in the subroutine DET, which evaluates the determinant. If four elements from the matrix have the configuration

$$\begin{matrix} a & b \\ c & d \end{matrix}$$

and c is to be set to zero by row reduction, d will be replaced by a value, x, as follows:

$$x = d - bc/a.$$

If d is located on the diagonal, complete loss of accuracy is checked for by computing

$$s = |x^2| \big/ |d^2|.$$

If s is less than 10^{-34}, x is not used; instead, d is replaced by 10^{-17}d. Note that this substitution will occur when x is zero, thus preventing zeroes on the diagonal. The power of ten, −17, is chosen to be near the total word size of 18 decimal places.

The above substitution prevents sudden jumps in the value of the determinant when all precision is lost at one step in the evaluation. This is important for the mode search routine which detects roots by looking for minima in a series of values of the determinant while one parameter is incremented slowly. A sudden jump will often produce a relative minimum which will be falsely interpreted as a root. At true roots, one or more elements along the diagonal are small, but not as small as those checked for here.

REFLECTION COEFFICIENTS AND OTHER AUXILIARY OUTPUTS

Once the depth functions of a mode have been determined, it is relatively easy to compute reflection coefficients at any interface. Therefore, a subroutine called RCOEF has been added to the program which will compute and print out reflection coefficients if requested by the use of control key 3. If key 3 is set to 1, the reflection coefficients at all interfaces are computed. If set to a number, n, greater than 1, the coefficient is computed at the *nth* interface only, where the surface is the first interface.

The printout includes the phase as well as the amplitude of the reflection coefficient and the grazing angle. The grazing angle, θ, of the equivalent rays is computed from the mode phase velocity and the sound speed, c, at the bottom of the layer, by Snell's law:

$$\theta = \cos^{-1} (c/v).$$

The grazing angle is computed only if the phase velocity is greater than the sound speed at the interface, since otherwise the equivalent ray does not reach the interface.

The reflection coefficient is derived, following Bucker (ref 9), by assuming that an isospeed layer exists for a small depth just above the interface. In this layer the depth function can be written as

$$f(z) = Ae^{ilz} + Be^{-ilz}, \tag{43}$$

where l, the vertical component of the mode wave number, is given for mode n by

$$l_n^2 = k_i^2 - \lambda_n^2 \tag{44}$$

and

$$k_i = \omega/c_{bi},$$

where c_{bi} is the sound speed at the bottom of layer i. The derivation now consists of identifying A and B as the pressures of the upgoing and downgoing waves at the bottom of the layer; thus the reflection coefficient

$$R = A/B.$$

A and B are evaluated by making f and its derivative at the interface between this small isospeed layer and the regular profile continuous with the normal mode depth functions. The thickness of the isospeed layer is then allowed to approach zero, giving the desired value of R. If F and F′ are the normal mode function and its depth derivative at the interface depth defined by eq (9) and (19), the reflection coefficient resulting from the above derivation is as follows:

$$R = (ilF + F')/(ilF - F'). \tag{45}$$

This coefficient is a complex number. Loss per reflection is given by 20 times the log of the absolute value. The phase gives the phase shift that an equivalent ray would

---

9. Sound Propagation in a Channel with Lossy Boundaries, by HP Bucker; J Acoust Soc Am, vol 48, p 1187-1194, November 1970.

experience upon reflection. Figure 3 is an example of the use of this computation. It shows phase and amplitude of the reflection coefficient in shallow water over a sandy-silt sediment lying over rock. The frequency is 1500 Hz. Reflections are given only at discrete points determined by the individual modes.

The model in figure 3 is for a liquid bottom. That is, no rigidity is supplied in this program and the sound speed, density, and attenuation determine the reflection coefficients.

The reflection coefficients computed by eq (45) can be closely approximated by dividing the mode attenuation by the loop length of the corresponding ray. The loop length must be determined from ray theory for the ray of the same phase velocity or vertexing velocity. However, an interesting analog of the ray loop length is the intermode interference length. This is discussed by Guthrie (ref 10). Specifically, if the difference between eigenvalues, $\mathrm{Re}\,\lambda_i$, for two adjacent modes is $\Delta\lambda$, the interference length $l = 2\pi/\Delta\lambda$. This distance will usually equal the ray loop length for some ray with phase velocity between that of the two modes.

As each mode after the first is computed, the length, l, is computed and printed out. Also routinely printed out for each mode is the mode damping or mode attenuation coefficient, in units of dB per km. This attenuation, α_i, is computed from the relationship

$$\begin{aligned}\alpha_i &= -1000\ \mathrm{Im}\ \lambda_i \log_{10} e \\ &= -8686\ \mathrm{Im}\ \lambda_i.\end{aligned}$$

This quantity multiplied by range gives the damping of mode i, in dB.

---

10. The Connection Between Normal Modes and Rays in Underwater Acoustics, by KM Guthrie; J of Sound and Vibration, vol 32, no 2, p 289-293, 1974.

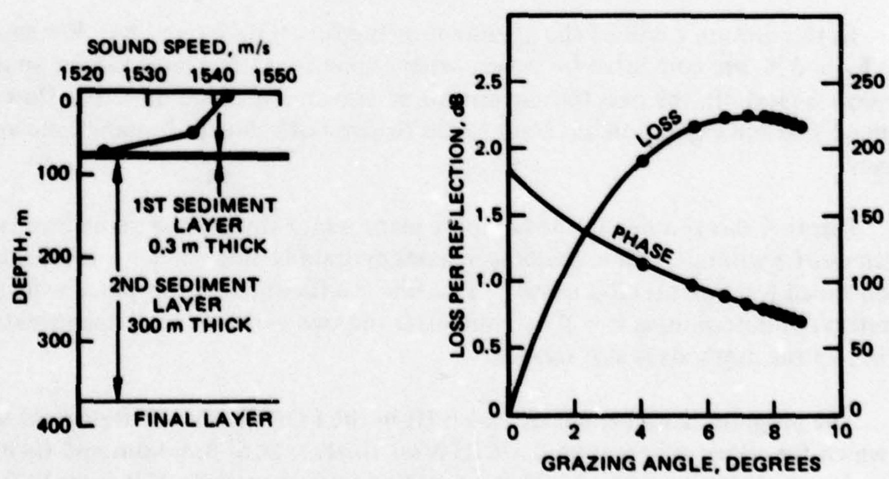


Figure 3. A shallow-water profile with resulting phase and amplitude of the reflection coefficient at 1.5 kHz. Parameters at the top of the sediment layers are as follows: 1st layer – c = 1606.45 m/s, $\gamma = 1.5\mathrm{s}^{-1}$, α = 0.18 dB/m, ρ = 1.68; 2nd layer – c = 1684.0 m/s, $\gamma = 1.5\mathrm{s}^{-1}$, α = 1.10 dB/m, ρ = 1.91; final layer – γ = -0.1.

COMPUTATION OF THE MODIFIED HANKEL FUNCTIONS

Most of the computer time required to determine eigenvalues and compute depth functions is spent in evaluating the modified Hankel functions of order 1/3. For this reason, minimizing computer time in evaluating these functions is desirable. Gaining as many places of accuracy as possible is even more important. The average normal mode computation will have many modes that can be determined to far greater accuracy than is required to obtain 0.1 dB accuracy in the propagation loss. However, there are usually some and often many modes in which many places of accuracy are lost in evaluating the determinant. Therefore, maximum accuracy in the modified Hankel functions is required to extend the range of cases for which computations can be carried out successfully.

Optimization of the program is a function of the computer word length. The program given in this report is for the UNIVAC 1110 with 60 bits word length in double precision or 18.1 decimal places. This section gives the equations and computational techniques that are required to optimize this program for different computer word lengths. Complete details of the functions are given in reference 11.

The Airy functions Ai(Z) and Bi(Z) can be used instead of the modified Hankel functions h_1 and h_2. However, since h_2 is ideally suited to matching the boundary conditions at great depth as formulated in this normal mode program, h_1 and h_2 are used here. The relationship between them is as follows:

$$h_1(z) = k\,[\mathrm{Ai}(-z) - i\,\mathrm{Bi}(-z)] \tag{46}$$

$$h_2(z) = k^*\,[\mathrm{Ai}(-z) + i\,\mathrm{Bi}(-z)]\ , \tag{47}$$

where

$k = (3/2)^{2/3}\,(1 - i\sqrt{3}/3)$, and k^* is the complex conjugate of k.

In this section z will be the argument of the functions h_1 and h_2. For small values of $|z|$, h_1 and h_2 are computed by power series expansions. For large values, an asymptotic expansion is used. In the past the asymptotic series was expanded directly. However, a continued fraction expansion has been found to give both shorter running time and better accuracy.

Figure 4 shows a line in the complex plane which divides the plane into two parts. For values of z within the line, the power series method is used. When z is outside the line, the continued fraction method is used. This line is a function of computer word length, and the method of determining it will be given after the two methods have been treated. The accuracy of the methods is also treated.

The program has a parameter called IH in the FORTRAN call statement which controls which functions are computed. If IH is set to zero, both functions and their derivatives are computed. If IH is set to 1, only the functions are computed. If it is set to 2, only h_2 and its derivative are computed.

11. Tables of the Modified Hankel Functions of Order One-Third and their Derivatives, Harvard University Computation Laboratory; Harvard University Press, Cambridge MA, 1945.

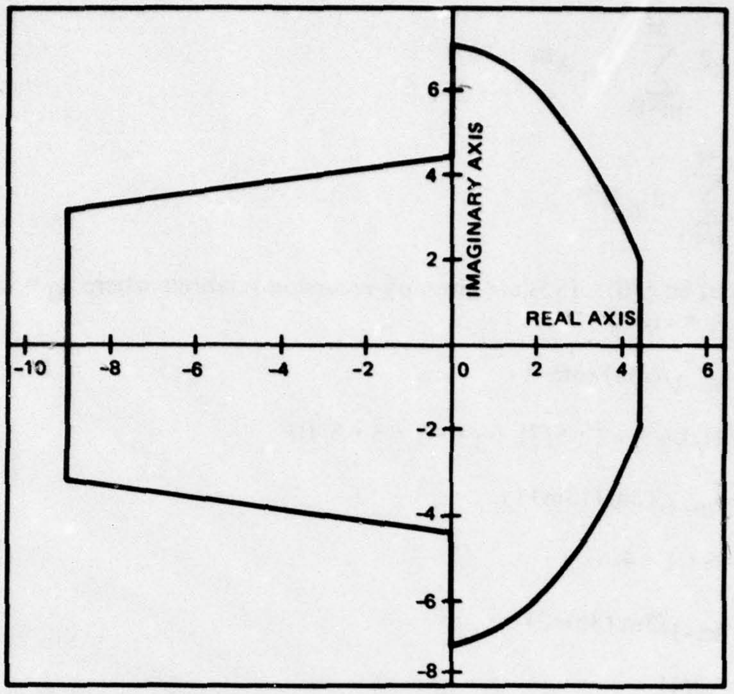


Figure 4. Line in complex plane dividing the arguments for which the modified Hankel functions are computed by (1) power series (inside) and (2) asymptotic expansion evaluated by continued fractions (outside).

POWER SERIES EXPANSION

In this expansion h_1 and h_2 are given by two auxiliary functions f and g as

$$h_1(z) = g + i(3)^{-1/2}(g - 2f) \tag{48}$$

$$h_2(z) = g - i(3)^{-1/2}(g - 2f) \ . \tag{49}$$

The auxiliary functions are given by the expressions

$$f = A \sum_{m=0}^{M} a_m X^m \tag{50}$$

$$g = Bz \sum_{m=0}^{M} b_m X^m \ , \tag{51}$$

where $X = z^3$, $A = 2^{1/3}/[\Gamma(2/3)]$ and $B = 2^{1/3}/[3^{2/3}\,\Gamma(4/3)]$. The derivatives $h_1'(z)$ and $h_2'(z)$ can be derived by straightforward differentiation of eq (50) and (51) to give

$$f' = -A z^2 \sum_{m=0}^{M} c_m X^m \tag{52}$$

$$g' = B \sum_{m=0}^{M} d_m X^m \; . \tag{53}$$

The coefficients of eq (50) – (53) are given by recursion relations where $a_0 = 1$, $a_1 = 1/3!$, $a_2 = +1 \cdot 4/6!$, $a_3 = -1 \cdot 4 \cdot 7/9!$

$$a_m = -a_{m-1}/(3m)(3m-1) \tag{54}$$

$b_0 = 1$, $b_1 = -2/4!$, $b_2 = +2 \cdot 5/7!$, $b_3 = -2 \cdot 5 \cdot 8/10!$

$$b_m = -b_{m-1}/(3m)(3m+1) \tag{55}$$

$c_0 = 3/3!$, $c_1 = -6 \cdot 1 \cdot 4/6!$

$$c_m = -c_{m-1}/3m\,(3m+2) \tag{56}$$

$d_0 = 1$, $d_1 = -4 \cdot 2/4!$

$$d_m = -d_{m-1}/3m\,(3m-2) \tag{57}$$

It is important for efficient computation that the number of terms M be no larger than necessary. In the current program the same value of M is used in all four sums. This is done because the optimum number never differs by more than one in the four cases and the determination by table look-up of four M's often would take longer than computing any unnecessary terms. M for each series is determined so that adding additional terms will not change the answer. Then the most stringent of the four conditions is tabulated and used.

A precise determination of the number of terms to use requires a knowledge of the size of the largest single term in the sum. When a term is smaller than this by a factor which is the power of 10 equal to the number of decimal places in the computer word size, it cannot affect the sum. We ignore the fact that a sum of small terms might be significant. This, then, defines the truncation point. Let m be the number of the largest term in the sum, k the number of terms to be used, and h the number of decimal digits in the machine word. Then for a given k, the largest absolute value of the argument z that can be used to compute g′ is given as

$$|z^3|^m d_m = |z^3|^k d_k \cdot 10^h \; . \tag{58}$$

The power of ten can be replaced by 2 raised to a power of the number of binary bits in the computer word if preferred. The coefficient d of eq (57) is used. Each of the other three should also be tried, to find the smallest number of the group for a given k. Equation (58) can be solved for |z|, giving

$$\log |z| = (\log d_k - \log d_m + h)/(3m - 3k) \; . \tag{59}$$

A simple computer program given in appendix C will find $|z|$ for each value of k from 1 up to the maximum number of terms desired. The largest term, m, is easily determined because from one k to the next m will remain the same or increase by 1, so it is only necessary at each step to check term m+1 to see if it is larger than term m.

The FORTRAN subroutine HANKEL given in appendix A uses the above power series method to compute h_1 and h_2 for small arguments. The coefficients a, b, c, and d are given in lists by that name. The truncation points are given in the list called ZMLA2, which lists values of $|z|^2$ determined by eq (59) or the three similar equations.

ASYMPTOTIC SERIES EXPANSION USING CONTINUED FRACTIONS

When the argument z falls outside the curve in figure 4, h_1 and h_2 can be computed more efficiently or more accurately by asymptotic series than by power series methods. Reference 11 gives information on branch cuts and regions of validity of the two forms of the asymptotic solution (Stokes' phenomenon). Here we will give computing formulas that comply with these requirements, without discussing them further.

Since a given expansion is valid in one or more quadrants, we choose complete quadrants as regions. For z in quadrants 1, 3, or 4 use

$$h_2(z) \sim \exp(5\pi i/12)\, F_2(z) \tag{60}$$

$$h_2'(z) \sim \exp(-\pi i/12)\, G_2(z) \tag{61}$$

For z in quadrant 2 use

$$h_2(z) \sim \exp(5\pi i/12)\, F_2(z) + \exp(11\pi i/12)\, F_1(z) \tag{62}$$

$$h_2'(z) \sim \exp(-\pi i/12)\, G_2(z) + \exp(-7\pi i/12)\, G_1(z) \tag{63}$$

For z in quadrants 1, 2, or 4 use

$$h_1(z) \sim \exp(-5\pi i/12)\, F_1(z) \tag{64}$$

$$h_1'(z) \sim \exp(\pi i/12)\, G_1(z) \tag{65}$$

For z in quadrant 3 use

$$h_1(z) \sim \exp(-5\pi i/12)\, F_1(z) + \exp(-11\pi i/12)\, F_2(z) \tag{66}$$

$$h_1'(z) \sim \exp(\pi i/12)\, G_1(z) + \exp(7\pi i/12)\, G_2(z) \tag{67}$$

The four auxiliary functions follow:

$$F_1(z) = K z^{-1/4} \exp(2i z^{3/2}/3) \sum_{m=0}^{M} C_M X^m \tag{68}$$

$$F_2(z) = k\, z^{-1/4} \exp(-2i\, z^{3/2}/3) \sum_{m=0}^{M} C_m Y^m \tag{69}$$

$$G_1(z) = k\, z^{1/4} \exp(2i\, z^{3/2}/3) \sum_{m=0}^{M} D_m X^m \tag{70}$$

$$G_2(z) = k\, z^{1/4} \exp(-2i\, z^{3/2}/3) \sum_{m=0}^{M} D_m Y^m \tag{71}$$

where X and Y equal $\mp i\, z^{-3/2}$ respectively, and

$$K = 2^{1/3}\, 3^{1/6}\, \pi^{-1/2} = 0.853\,667\,218\,838\,951$$

The coefficients C_m and D_m are again computed by recursion relations where $C_0 = D_0 = 1$:

$$C_m = C_{m-1}\, [9\,(2m-1)^2 - 4]/48m \tag{72}$$

and

$$D_m = D_{m-1}\, [9\,(2m-1)^2 - 16]/48m\ . \tag{73}$$

Square roots of z are to be taken so that the real part of the root is always positive and the imaginary part has the same sign as the imaginary part of z. This applies also to fourth roots. The three-halves power is obtained as the product of z and its square root.

The summations in eq (68) – (71) can be done as indicated or evaluated by continued fractions. When done as indicated they are asymptotic series, and care must be taken to truncate them at the term of smallest magnitude, if this term is reached, because adding more terms will reduce the accuracy. Since the largest term in these series will always be 1, the series can be truncated if the terms become less than 10^{-h} in magnitude, where h is the number of decimal digits in the computer word.

Continued Fraction Expansion

The method of continued fractions is more effective in evaluating these asymptotic series, and it is used in subroutine Hankel in the FORTRAN program in this report. The coefficients are stored in lists entitled C4, C5, D4, and D5. In the remainder of this section the continued fraction technique is presented, along with the method of determining coefficients.

The continued fraction has the form

$$F(x) = b_0 + \cfrac{a_1}{x + b_1 + \cfrac{a_2}{x + b_2 + \ldots}}\ . \tag{74}$$

It is to be used to evaluate a polynomial

$$P(x) = \sum_{m=0}^{M} C_m X^m \quad . \tag{75}$$

This polynomial can represent any of eq (68) – (71). One of three standard forms for continued fractions, this form is used because it has two coefficients at each stage and therefore is equivalent to an asymptotic series of twice as many terms. This reduces by half the number of divisions required. Since complex divisions are lengthy, requiring six real multiplications and two divisions, this is the only standard form of the continued fraction that can compete in computer time with the asymptotic series.

The coefficients a_i and b_i of eq (74) must be determined from the coefficients C_m. The usual technique is to express P as a rational function, then use the continued fraction to evaluate the rational function. The determination of the coefficients can be done in these two steps or by a second method which goes directly from power series to continued fraction coefficients. The second method is preferable because the loss of accuracy is more in the first. But since the first method is more easily understood, each method will be given; a computer program is included in appendix C which will determine coefficients by the second method.

Let M in eq (75) be an even number so that 2N = M. (An additional unnecessary term of the series can always be used.) The rational function will have the form

$$R(x) \approx k \sum_{i=0}^{N} \hat{e}_i x^i \Big/ \sum_{i=0}^{N} \hat{f}_i x^i \; , \tag{76}$$

where $\hat{e}_0 = \hat{f}_0 = 1$ and $k = C_0$. The coefficients $\hat{e}_i$ and $\hat{f}_i$ are evaluated from a set of linear equations which can be described by displaying a particular case. For N = 3 they are as follows:

$$\begin{bmatrix} -1 & 0 & 0 & C_0 & 0 & 0 \\ 0 & -1 & 0 & C_1 & C_0 & 0 \\ 0 & 0 & -1 & C_2 & C_1 & C_0 \\ 0 & 0 & 0 & C_3 & C_2 & C_1 \\ 0 & 0 & 0 & C_4 & C_3 & C_2 \\ 0 & 0 & 0 & C_5 & C_4 & C_3 \end{bmatrix} \begin{bmatrix} k\hat{e}_1 \\ k\hat{e}_2 \\ k\hat{e}_3 \\ \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \end{bmatrix} = - \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{bmatrix} \tag{77}$$

With e_i and f_i thus determined, R(x) is equivalent to P(x) through the first M + 1 terms. R(x) can now be evaluated exactly (except for round-off error) using a continued fraction of the form F(x) of eq (74).

Rather than $R(x)$, however, a similar expression in $y = 1/x$ is the form that is well suited to evaluating asymptotic series. This expression is obtained by dividing each term of $R(x)$ by X^N. The order of the coefficients can now be reversed and a simple algebraic operation can yield a value of 1 for each of the two initial coefficients and a new value for k. We will call this new rational function with renamed coefficients $\hat{R}(y)$. It will have the form of eq (76) but different coefficients, say e and f instead of $\hat{e}$ and $\hat{f}$.

The coefficients a_i and b_i are determined from e_i and f_i by a recursive formula which involves constructing an $n \times n$ triangular matrix Q with elements $q_{i,j}$ as follows:

$$b_0 = e_0$$

$$q_{1,i} = (e_i - e_0 f_i)/a_1 \qquad i = 1, 2, \ldots, N ,$$

where $q_{1,1} = 1$, giving a_1, and

$$b_1 = f_1 - q_{1,2} .$$

The second row:

$$q_{2,i} = (f_i - q_{1,i+1} - b_1 q_{1,i})/a_2 \qquad i = 2, 3, \ldots, N ,$$

where $q_{2,2} = 1$, giving a_2, and

$$b_2 = q_{1,2} - q_{2,3} .$$

Elements outside the matrix are assigned a value of zero. The remaining rows for m = 3 to N are as follows:

$$q_{m,i} = (q_{m-2,i} - q_{m-1,i+1} - b_{m-1} q_{m-1,i})/a_m \qquad i = m, m+1, \ldots, N ,$$

where $q_{m,m} = 1$, giving a_m, and

$$b_m = q_{m-1,m} - q_{m,m+1} .$$

The second method determines the continued *fraction* coefficients a_i and b_i directly from the asymptotic series coefficients c_i. This method is preferable to the first because the loss of accuracy in inverting the matrix in eq (77) can be more than the loss in this second method.

It has been pointed out* that the second method is probably a variant of the Viskovakoff algorithm described by Khovanskii (ref 12) and as such is unstable – subject to accumulation of errors. However, it is sufficiently stable to obtain the required coefficients.

---

*Private communication with AN Stokes, CSIRO, Wembley, Western Australia.

12. The Application of Continued Fractions and their Generalizations to Problems of Approximate Analysis, by AN Khovanskii; a monograph in Russian, 1956.

The coefficients are derived as follows. The well-known recursive relations that give the N*th* stage of a continued fraction as a rational function are used (ref 13).

$$F_N(y) = A_N(y)/B_N(y) \ , \tag{78}$$

where

$$A_N = \sum_{i=0}^{N} e_i y^i$$

$$= (y + b_N) A_{N-1} + a_N A_{N-2} \tag{79}$$

$$B_N = \sum_{i=0}^{N} f_i y^i$$

$$= (y + b_N) B_{N-1} + a_N B_{N-2} \ , \tag{80}$$

in which $A_{-1} = 1$, $A_0 = b_0$, $B_{-1} = 0$, and $B_0 = 1$. Again $y = 1/x$. The long division indicated in eq (78) is then carried out, giving a quotient in terms of a_i, b_i, and y that can be equated, term by term, to the first 2N−1 terms of the asymptotic series.

The long division is carried out with A_N and B_N written in descending powers of y. The quotient is then in descending powers of y or ascending powers of x. Fortunately, the first 2N+1 terms determined for any N are identical to the same initial terms for any larger value of N. This will be proven later. The first few equations obtained from the division are as follows:

$$\begin{aligned} b_0 &= C_0 \\ a_1 &= C_1 \\ -a_1 b_1 &= C_2 \\ a_1 \left(b_1^2 - a_2\right) &= C_3 \\ a_1 \left(2 a_2 b_1 - b_1^3 + a_2 b_2\right) &= C_4 \end{aligned} \tag{81}$$

From these equations a_i and b_i can be determined, since the coefficients C_i are known. However, a simpler method is available.

The long division indicated in eq (78) can be carried to 2N+1 valid places; but beyond N+1 places, terms from the original dividend are no longer entering the remainder. Therefore terms in the later part of the quotient have a simplified form. Since term n+1 of

13. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, ed by M Abramowitz and IA Stegun; National Bureau of Standards Applied Mathematics Series, vol 55, p 19, 1964.

the quotient is equal to the *nth* asymptotic coefficient, designate it C_m. Note that the C's are numbered from 0 to N. Let the coefficient of y^m in B_n be $B_{n,m}$. Then

$$C_{N+j} = - \sum_{i=1}^{N} B_{N,N-i} C_{N+j-i} , \qquad 1 \leq j \leq N . \tag{82}$$

Here the C's are numerical constants. The unknowns, the a's and b's, are in the terms of B. Suppose that these unknowns have been determined up to n = N - 1. Then eq (82) will contain two unknowns, a_N and b_N. By using eq (82) for j = N - 1 and N, the unknowns can be evaluated. The index N can then be increased by 1 and the process repeated. The process can start with N = 2 if a_1, b_0, and b_1 are provided, but these are easily determined from eq (81). The terms of B_N are determined from eq (80), which gives for each term

$$B_{n,m} = B_{n-1,m-1} + b_n B_{n-1,m} + a_n B_{n-2,m} . \tag{83}$$

Any $B_{n,m}$ is zero if m is greater than n.

When j = N - 1 is used in eq (82) in the process described above, the coefficient of the (2N-1) power of x is being evaluated. This term is expected to contain a_N and b_N, but – as will be proven later – because the coefficient of b_N is zero, a_N is the only unknown in a linear equation and can be easily evaluated. The next term determined with j = N contains a_N and b_N, but now only b_N is unknown and is easily evaluated.

As an example, the C_n's through n = 10 are listed in table 1. These are the asymptotic series coefficients given by eq (72). The corresponding a_n's and b_n's as determined above are also listed. A more complete list of the a's and b's can be obtained from the FORTRAN program in appendix C.

Table 1. Asymptotic series coefficients, C_n, and the corresponding continued fraction coefficients, a_n and b_n.

n	c_n	a_n	b_n
0	1.	0	1.
1	0.10416	0.10416	-0.80208
2	0.08355	-0.58764	-2.28555
3	0.12823	-2.29072	-3.77864
4	0.29185	-5.11525	-5.27462
5	0.88163	-9.06285	-6.77193
6	3.32141		
7	14.99576		
8	78.92301		
9	474.45154		
10	3207.49009		

A FORTRAN program to compute the continued fraction coefficients for the series given by eq (72) is given in appendix C. This program can be easily modified to determine the other set.

Two Proofs

In this section proofs will be given of two facts used in the previous section. Following this, the number of terms required, the accuracy, and similar topics will be discussed. To prove that the first 2N+1 terms of the quotient A_N/B_N are equal to the same terms when N is a larger integer, use long division on eq (79) and (80) to obtain

$$A_N/B_N = A_{N-1}/B_{N-1} + a_n\,(A_{N-2}\,B_{N-1} - A_{N-1}\,B_{N-2})/(B_N\,B_{N-1}) \ . \tag{84}$$

If the first quotient on the right is to have terms equal to the quotient on the left up through term 2N-1, the remainder must have no terms with y to a higher power than -(2N-1). The final divisor, $B_N\,B_{N-1}$, contains y to the (2N-1) and lower powers. Therefore, the proof is complete if the numerator of the remainder is a constant. To show this, use eq (79) and (80) to evaluate B_{N-1} and A_{N-1}; it can be shown that

$$\begin{aligned} A_{N-2}\,B_{N-1} - A_{N-1}\,B_{N-2} &= -a_{N-1}\,(A_{N-3}\,B_{N-2} - A_{N-2}\,B_{N-3}) \\ &= (-1)^N\,a_{N-1}\,a_{N-2}\cdots a_1 \ . \end{aligned}$$

The right-hand product is obtained by repeatedly applying the middle result. The product of a's is a constant, completing the proof.

The second proof required is that in the quotient of A_N/B_N, C_{2N} (the coefficient of y^{-2N}) will contain no a_i or b_i to higher than term N and C_{2N+1} (the coefficient of y^{-2N-1}) will involve no a_i to higher than term N+1 and no b_i to higher than term N.

The first part is intuitively obvious. Since from the preceding proof C_{2N} will be the same when derived from the ratio A_x/B_x for any x as long as it is N or greater, we need consider only the case where x is N. But since from eq (79) and (80) A_N and B_N contain no a's or b's of greater than term N, C_{2N} cannot contain a's or b's of higher terms.

By the same argument C_{2N+1} can contain no a's or b's to higher terms than N+1. There remains to be proven only that b_{N+1} cannot exist in C_{2N+1} or that its coefficient, which we will call E, is zero. Applying eq (82) for N+1 and j = N gives

$$C_{N+1+N} = -\sum_{i=1}^{N+1} B_{N+1,N+1-i}\,C_{N+1+N-i} \ .$$

E, the coefficient of b_{n+1} in this expression from eq (83), takes the form

$$E = -\sum_{i=1}^{N+1} B_{N,N+1-i}\,C_{2N+1-i} \ .$$

But by choosing $j = N$ in eq (82) we see that terms 1 to N for C_{2N} equal terms 2 to N+1 in E, so

$$E = -B_{N,N}C_{2N} + C_{2N} .$$

However, since $B_{N,N}$, the coefficient of y^n in B_n, is always 1 by eq (80), $E = 0$. Therefore b_{N+1} does not exist in C_{2N+1}.

Number of Terms

The number of terms or stages to use in the continued fraction was arrived at by a trial and error process. For a given number of terms, a real positive argument was decreased until the accuracy began to drop. The magnitude just before this drop was considered to be the optimum point to increase the number of stages by one. Because the argument to the continued fractions is $z^{3/2}$, we took the larger of the magnitudes of the real and imaginary parts of $z^{3/2}$ as the test number. This number is then compared to the 3/2 power of the points determined along the real axis by trial and error.

The above method appears to work well although it involves no thorough understanding of the way complex numbers affect the successive convergents of a continued fraction. Table 2 shows the points down to which a given number of stages gives full accuracy for positive real arguments and lists the 3/2 power of these numbers as used in the FORTRAN program list called ZMLA5.

Division Lines

The power series method is now to be used for small arguments and the continued fraction method for large arguments. The exact dividing line between them is needed. The division line of figure 4 was arrived at by computing the functions along rays from the origin, using both power series and continued fractions. The number of decimal places to which the functions determined by the two methods agree tends to reach a maximum at some distance from the origin along each ray. At distances short of this maximum we can assume that the continued fraction method is less accurate than the power series. At distances beyond the maximum, the power series is assumed to be less accurate. The maximum therefore indicates the ideal place to change from one method to the other if the decision is to be based solely on accuracy. This method was used to determine figure 4.

A complication arises, however. Along certain rays from the origin, h_1 and its derivative reach a maximum number of places at very different distances from h_2 and its derivative. The principal problem is at ±60° but persists from about 30° to 90°. At 60°, h_1 is small in magnitude and h_2 is large. The power series method cannot compute the small values accurately due to loss in accuracy in subtraction in eq (48). The accuracy of the continued fraction for h_2 is poor at 60° because eq (69) becomes a nonalternating series and continued fraction approximations are not known to improve the accuracy of nonalternating asymptotic series as they do for alternating series.

A reasonable solution to this problem is to compute h_1 by continued fractions and h_2 by power series for arguments at these angles and magnitudes from 4 to 10. However, as will be shown later, the above solution has not been employed at this time since this area is not of great importance for normal mode computations. Instead, the argument was chosen

Table 2. Cut-off points for determining the number of stages in the continued fractions.

Number of Stages	Real Argument x	Program Test Value $x^{3/2}$
1	10^6	10^9
2	80	715.0
3	35	207.0
4	22	103.0
5	13	47.0
6	11	36.4
7	9	27.0
8	8	22.6
9	7	18.5
10	6.5	16.6
11	6	14.7
12	5.8	14.0
13	5.5	12.9
14	5.3	12.2
15	5.1	11.5
16	4.9	10.8
17	4.5	9.5
18	4.4	9.2

that gave equivalent accuracy for the two methods. Along 60° this minimum accuracy is 9 decimal places.

The following relationship exists between h_1 and h_2 for positive and negative values of the imaginary part of the arguments:

$$h_1(z^*) = [h_2(z)]^* ,$$

where the * means complex conjugate. Thus, the above discussion at 60° can be translated to −60°. Also, the functions actually need to be computed only in quadrants I and II. They could then be evaluated in quadrants III and IV by the above relationship. The above relationship explains the symmetry of figure 4 about the real axis.

COMPARATIVE ACCURACY

The accuracy of the three methods – power series, asymptotic series and continued fractions – has been determined on a CDC computer with 48 bits or 14.4 decimal places of accuracy in the floating point word. Since this differs from the double precision word length of 60 bits or 18.1 decimal places that applies to the preceding part of this report, these results are for comparative and illustrative purposes only.

Accuracy is determined by computing the functions and either comparing the answers for the several different computing methods or computing the wronskian. The wronskian is a constant given by the relationship

$$h_1 h_2' - h_2 h_1' = -1.457495441 04i = -i\,96^{1/3}/\pi \,. \tag{85}$$

The wronskian will determine the accuracy of the functions if it can be computed without loss of accuracy. If the two products in it are large, though, accuracy will be lost in the subtraction. This generally happens for arguments near the negative real axis. Here accuracy must be determined by comparing answers from different methods. The accuracy of the functions and their derivatives will generally be about equal.

Figure 5 illustrates the accuracy that is obtained in different parts of the complex plane of the argument, z, by using the power series method. On the inner contour, the functions h_1 and h_2 and their derivatives have 12 places of accuracy. On the outer contour, the accuracy is 11 places. As expected, the accuracy is best for arguments of small magnitude. The accuracy remains best in directions from the origin in which the functions are large in magnitude. This is because less accuracy is lost in subtraction. Accuracy must be lost when individual terms of the series are large but the sum is small.

Figure 6 shows accuracy contours for the asymptotic expansion with both the direct and continued fraction evaluation of the series. Here, the best accuracy is obtained for large arguments, and accuracy decreases toward the origin. As can be seen, each of the two methods is better in some directions from the origin. The choice of methods then depends upon which directions are of most value to the normal mode program. The dots on the figure show the locations at which the functions were evaluated in a typical surface duct run. Although arguments can lie anywhere in the plane, most of them follow this pattern. They lie just above the negative real axis and in a narrow angle above the positive real axis. The continued fraction method is distinctly better on this positive side. Since computing time also favors the continued fraction method, it is clearly the method to use.

If the 12-place accuracy contour from figure 6 lies inside that for figure 5 at some angle from the origin, 12 places can be obtained at any range along this angle by using either power series or asymptotic expansion in the interval of overlap. If the asymptotic expansion contour lies outside the other, there is an interval in which 12 places cannot be obtained. Only some lesser number of places can be obtained in this interval. These contours apply when both functions and their derivatives are all computed by a single method. As mentioned earlier, increased accuracy could be obtained in some areas by computing the two functions by different methods.

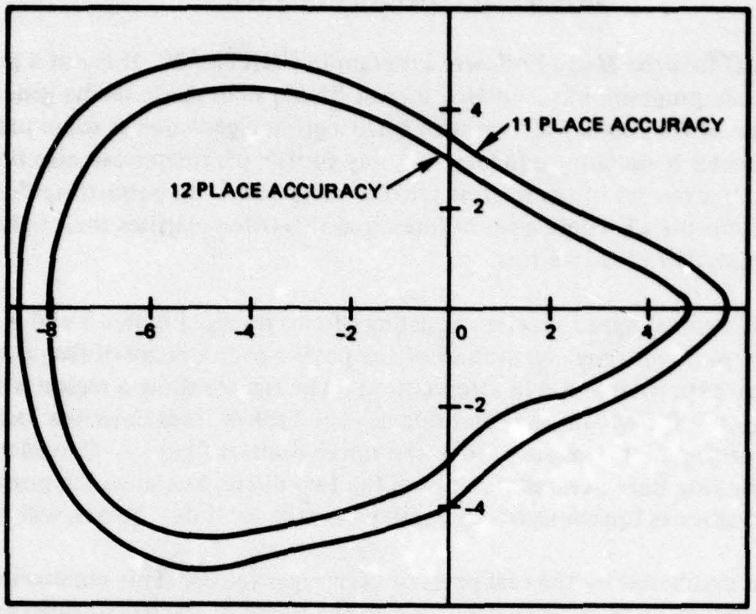


Figure 5. Locus of arguments for which the power series evaluation of the modified Hankel functions gives 12 and 11 decimal places of accuracy for a computer word length of 14.4 decimal places.

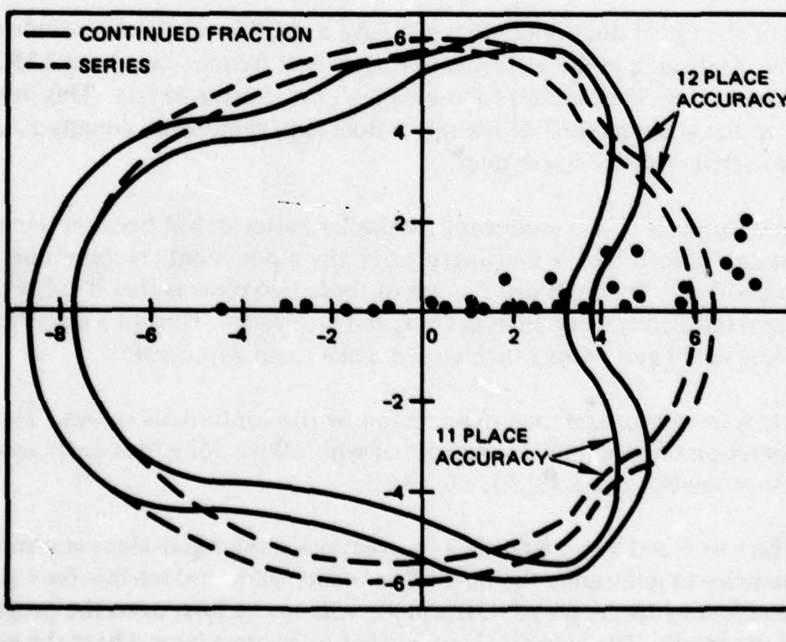


Figure 6. Locus of arguments for which the direct and continued fraction evaluation of the asymptotic series gives 11 and 12 decimal place accuracy. The arguments at which the modified Hankel functions were evaluated in a typical normal-mode run are shown.

MODE FOLLOWER PROGRAM

Appendix D lists the Mode Follower Program in FORTRAN. It is not a part of the general normal mode program, but is related in that it uses some parts of the general program. The purpose of the mode follower is to trace a given eigenvalue as some parameter is varied. This parameter is usually frequency, but any profile parameter can also be varied. The eigenvalues at a given set of parameters are discrete points. By permitting the parameter to vary, the eigenvalues become a set of lines, and this often clarifies their behavior at the fixed points. Figures 7-9 illustrate this.

Figure 7 is a sound speed profile consisting of two ducts. Figures 8 and 9 show the real and imaginary parts of some eigenvalues of the profile over a range of frequencies. The imaginary parts are expressed as mode attenuations. The figures show a region where both ducts are exerting an influence on the eigenvalues. The broken lines show the location of eigenvalues for a profile that consists of only the upper duct of figure 7. Considerable time could be spent studying the interaction between the two ducts, but since the purpose here is to illustrate eigenvalues as functions of a parameter, only a brief description will be given.

Modes are numbered by the real parts of their eigenvalues. This numbering is consistent with the number of beats or changes of π in the phase of the depth functions. Thus the eigenvalue of a mode numbered 1 in a profile consisting of only the upper duct lies exactly over the eigenvalues of a mode in the double duct in figures 8 and 9, but this mode in the double duct changes number each time it crosses the real part of another mode. The depth function actually gains an additional beat each time this happens. The background of modes that are being crossed consists of the higher order, untrapped modes associated with the lower duct.

Mode 2, of the upper duct only, does not have a single mode in the double duct that overlies it exactly. Instead, a mode attempts to follow it at frequencies above 1350 Hz. Below this frequency, successive modes follow its path for short intervals. This interplay between modes occurs when mode 2 of the upper duct is in some sense equally as untrapped as the modes associated with the lower duct.

The imaginary parts of the modes follow similar patterns; but because the mode numbering is not determined by the imaginary parts, the mode numbers sometime jump from one line to another. An important feature of these two plots is that if the real parts of the eigenvalues cross, the imaginary parts do not; and vice versa. Thus two eigenvalues do not tend to become equal at a point which would make them degenerate.

The mode follower program will tend to follow the continuous curves. Thus if started in the right direction on mode 59 at 1450 Hz, it will follow along the continuous mode which becomes successively mode 58, 57, 56, and 55.

Figures such as 8 and 9 can be drawn by computing the eigenvalues at a sufficient number of frequencies to determine the lines. The mode follower does this for a given eigenvalue while adjusting the step size so the mode will not be lost, or so the program will correctly follow the mode. The step size is permitted to become large where the eigenvalue can be approximated by a parabolic curve, but it shortens when extrapolation to the next point becomes less accurate.

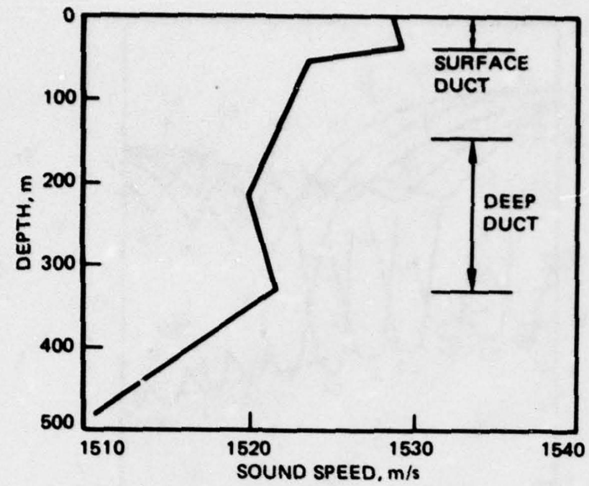


Figure 7. A five-layer approximation to the sound speed of a surface duct overlying a refractive duct.

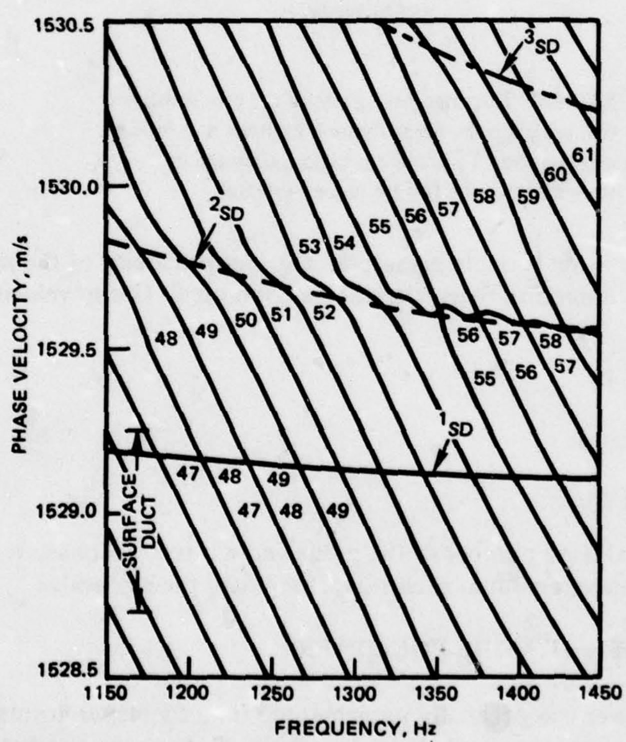


Figure 8. The real part of the eigenvalues as continuous functions of frequency for the profile of figure 7. Some mode numbers are given. The first three modes for the surface duct only (SD) are shown as broken lines. That for mode 1 coincides with an existing line.

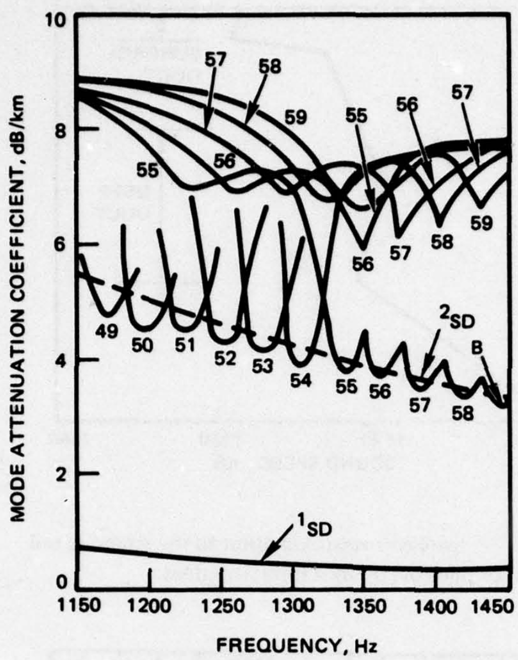


Figure 9. The imaginary parts of the modes whose real parts are shown in figure 8 expressed as mode attenuation. To avoid confusion, they are not shown across the full frequency interval.

When frequency is the variable parameter, the group velocity of the mode can be computed easily since a numerical derivative can be computed. Group velocity is given by the relationship

$$C_g = d\omega/d\,(\mathrm{Re}\,k)$$

$$\cong \Delta\omega/\Delta\,(\mathrm{Re}\,k)$$

$$\cong -\Delta f\, v^2/f\,\Delta v \ ,$$

where k is the horizontal wave number of the mode and v is the real phase velocity. The mode follower prints this value out at each step, along with the eigenvalue.

IMPLEMENTATION OF THE MODE FOLLOWER

The mode follower was originally implemented for a two-layer normal mode which differed from the n-layer program in that the derivative dG/dv of the characteristic equation was evaluated along with G. The iteration for roots of G was thus Newton-Raphson and is given by the relationship

$$v_{i+1} = v_i - G/G' \ . \tag{86}$$

This is simpler than the secant iteration of eq (15), in which G′ must be evaluated numerically. Because of the simpler iteration, an effective scheme for mode following was available.

Since considerable effort was required to develop a similar scheme for the n-layer case, the two-layer mode follower will be briefly described to serve as an introduction to the n-layer case.

The two-layer mode follower employs one iterative step of eq (86) at each point where G is evaluated. Thus, a root that is inexact but sufficiently exact is obtained. The original estimate is obtained by extrapolating from the three most recent roots. If this estimate is sufficiently close to the true root, the single iterative step will make a small correction, G/G', that will bring the estimate very close to the true root. By using the size of this correction to control the step size, the program is self-regulating. The program works well when a permissible value of G/G' of 10^{-6} to 10^{-4} is used. Outside this interval the step size is either doubled or halved.

The multiple-layer program differs from this in several details. The extrapolation from the previous three points is done not only for the phase velocity but also for the numerical derivative,

$$D^{-1} = \Delta v/\Delta G.$$

Lagrange three-point interpolation is used, given by the form

$$v(x) = \frac{v(x_1)(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} - \frac{v(x_2)(x-x_1)(x-x_3)}{(x_1-x_2)(x_2-x_3)} + \frac{v(x_3)(x-x_1)(x-x_2)}{(x_1-x_3)(x_2-x_3)} , \tag{87}$$

where x is the new value of the parameter that is being varied (usually frequency) and x_1, x_2, and x_3 are the three previous values, x_1 being the most recent. To extrapolate the derivative, v is replaced by D^{-1} in eq (87). Both quantities are complex numbers.

The determinant is now evaluated at this new phase velocity to give a value G_0. Next a corrected value of phase velocity, v_0, is obtained:

$$v_0 = v - G_0 D^{-1} . \tag{88}$$

In the two-layer case, the size of the correction, GD^{-1}, is used to control the step size. Because the numerical derivative is less precise, we evaluate G once more at this new position, obtaining G_1. A new numerical derivative is next calculated:

$$D_0^{-1} = (v_0 - v)/(G_1 - G_0) .$$

This derivative is now compared with the extrapolated value to determine whether the step size should be changed. To do this an error

$$E = |1 - D_0/D|^2$$

is computed. Good results have been obtained by keeping E between 10^{-5} and 10^{-2}. If E becomes larger than this, the step size is halved and the extrapolation is tried again. Should halving the step size five times fail to obtain a value of E less than 10^{-2}, the mode is presumed to be lost and the program halts.

If E is less than 10^{-2}, the step is successful and the stored values are updated for the next step. Before v is stored, though, the iterative step of eq (88) is applied one more time to obtain a more precise value of v. This requires little extra effort because the quantities G_1 and D_0^{-1} are already available.

If the error E is less than 10^{-5}, the next step size is doubled.

It is possible for the extrapolation to be too successful. That is, if v is very near the true root, G_0 and Δv will be very small. The numerical derivative may then be inaccurate. Therefore, before the error term E is computed, a quantity

$$F = |v/\Delta v|^2$$

is computed. If F is greater than 10^{28}, the extrapolated derivative is used rather than the computed derivative and the program proceeds to the next step. If F is greater than 10^{34}, the step size is doubled before proceeding to the next step.

The other principal part of the program is the initialization which must evaluate v at three values of x to obtain the numbers needed for the first extrapolation, eq (87).

INPUT AND OUTPUT

The first input card contains the maximum number of steps allowed, the limits applied to E and F, and keys which control both the amount of detail in the printout and whether the profile parameters are to be read in or retained from the previous run. Default values are supplied when these items are left blank. Next the profile parameters are read in. These are an older style and only permit specification of the absorption loss at the top of a layer. The sound speed gradient is assumed to be real at the top of any layer.

A final card indicates which variable – frequency, sound speed, depth, gradient, absorption, or density – will be varied, by specifying a number called nx in the program, from 1 to 6. The next number, ny, specifies which layer the variable will be in. This layer number is not needed if frequency is selected. A third number, nz, indicates, if zero, that the profile will remain continuous as the selected parameter is varied. If nz is not zero, the selected parameter moves alone without a compensating motion in other profile parameters. The card next gives the initial and final value of the parameter to be varied and the initial step size. Finally, the particular mode to be followed is indicated by giving an approximate phase velocity and an initial step size. These must be chosen such that the subsequent iteration will converge on the correct mode.

The principal output of this program is the print statement at line 314. Each line of output contains the value of the parameter being varied, the complex phase velocity, the determinant G, the derivative D^{-1}, the error term E, the mode attenuation, the mode group velocity (if frequency is the parameter being varied), and the step number. After the final step, the profile in its final form is printed out.

CONCLUSIONS

1. An effective program for computing propagation loss in a layered ocean by normal modes has been developed. Complete documentation for the program is contained herein.

2. Sediment layers are modeled as fluids in which densities, sound speeds, and absorption can be specified. This permits a complete wave solution for bottom reflected sound energy.

3. A continued fraction technique for evaluating asymptotic series is shown to give superior results in evaluating the auxiliary functions required in this program, the modified Hankel functions of order 1/3.

4. A mode follower program given here is useful in tracing eigenvalues. Such traces are needed to understand the eigenvalue structure.

RECOMMENDATIONS

1. Improve the mode locating ability of this normal-mode program to make it self-contained. It currently requires user interaction to locate eigenvalues.

2. Investigate methods to incorporate the effect of rough boundaries into this program.

REFERENCES

1. The Bilinear Modified-Index Profile, by WH Furry, in Propagation of Short Radio Waves, DE Kerr, ed; MIT Rad Lab series, vol 13, p 140–168, McGraw-Hill, New York, 1951.
2. Navy Underwater Sound Laboratory Report 111, Theory of the Anomalous Propagation of Acoustic Waves in the Ocean, by HW Marsh, 1950.
3. Normal-Mode Theory Applied to Short-Range Propagation in an Underwater Acoustic Surface Duct, by MA Pedersen and DF Gordon; J Acoust Soc Am, vol 37, p 105–118, January 1965.
4. Naval Air Development Center Report NADC-72002-AE, Normal Mode Solutions and Computer Programs for Underwater Sound Propagation, by CL Bartberger and LL Ackler, 4 April 1973.
5. A Normal Mode Theory of an Underwater Acoustic Duct by Means of Green's Functions, by RL Deavenport; Radio Sci, vol 1, p 709–724, 1966.
6. Analytic Description of the Low-Frequency Attenuation Coefficient, by WH Thorp; J Acoust Soc Am, vol 42, p 270, 1967.
7. Some Effects of Velocity Structure on Low-Frequency Propagation in Shallow Water, by AO Williams; J Acoust Soc Am, vol 32, p 363–365, March 1960.
8. Sound Attenuation as a Function of Depth in the Sea Floor, by EL Hamilton; J Acoust Soc Am, vol 59, p 528–535, March 1976.

9. Sound Propagation in a Channel with Lossy Boundaries, by HP Bucker; J Acoust Soc Am, vol 48, p 1187-1194, November 1970.

10. The Connection Between Normal Modes and Rays in Underwater Acoustics, by KM Guthrie; J of Sound and Vibration, vol 32 no 2, p 289-293, 1974.

11. Tables of the Modified Hankel Functions of Order One-Third and their Derivatives, Harvard University Computation Laboratory; Harvard University Press, Cambridge MA, 1945.

12. The Application of Continued Fractions and their Generalizations to Problems of Approximate Analysis, by AN Khovanskii; a monograph in Russian, 1956.

13. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, ed by M Abramowitz and IA Stegun; National Bureau of Standards Applied Mathematics Series, vol 55, p 19, 1964.

APPENDIX A: NORMAL MODE PROGRAM IN FORTRAN

This program consists of the main program and seven subroutines. The main program handles the input and output and performs much of the computation. This includes profile preparation, mode search, determination of depth function coefficients, normalization, computation of depth functions, and summation of modes. Auxiliary functions are performed by the subroutines SETUP and DET, which set up the determinant, then evaluate it. This is the determinant from which eigenvalues are determined. The subroutine HZERO determines the Hankel functions of order zero, second type, which gives the range dependence of the modes. Only a single term of the asymptotic expansion is needed for this function.

Subroutine HANKEL evaluates the modified Hankel functions of order 1/3, by which the depth dependence of the modes is expressed. The majority of computing time is usually expended in this subroutine. Subroutine CFR is used by subroutine HANKEL to evaluate continued fractions. Subroutine RCOEF evaluates and prints reflection coefficients when they are requested.

```
C      THIS IS THE MAIN PART OF   NLAYNM
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       DOUBLE PRECISION  LAMBDA,LAMBDI
       INTEGER COL
       REAL R ATTEN, T RE, RX
       DIMENSION LOSPCH(5,25)
       COMMON /HAN/ H2R,H2I,H1R,H1I,H2PR,H2PI,H1PR,H1PI,R
       COMMON/INPUT/ Z(12), N, OMEGA, V, VI, CON(12), GSQ(12),
      1 CAY(12), LAMBDA, LAMBDI, G(12)
      2,RHO(12), GI(12), G SQI(12), CAYI(12)
       COMMON /EXPO/ EXSUM, CNTR, RATIO(25)
       COMMON/DETMNT/ A(25,4), Q(25,4)
       COMMON/PARTS/ ZT(12),ZTI(12),ZB(12),ZBI(12)
       COMMON/REFL/ AF(12,200), AG(12,200), BF(12,200), BG(12,200),
      2 EIGEN(350), EIGENI(350), B (25,4), BI(25,4), CB(12), CBI(12),
      3 CAYSQ(12), CAYSQI(12), NN
       DIMENSION D(350), DI(350), F(100), FI(100), HZERO2(350),
      * DA(350), SRES(350), GAMMAI(12), BLPK(12),
      4 HZER2I(350), DPK(12), GCU(12), GCUI(12), CI(12),
      3 PHASE V(350), PHASI V(350), UU(2000), UUI(2000)
       COMMON /LIMIT/ TLIM, EXPONT, SLIM
       DIMENSION LOSS(101)
       DIMENSION C(12), DEPTH(52), DBLOSS(350), COL(120),
      1CONTR(10), EF(2), FMAG(350), FANG(100),
      2GAMMA(12),JSMBL(10),JCOUNT(5),JCOU(5),LEVEL(41),PLEV(5),RLOSS(100)
      3 , RLOS(101),RECVRS(51),TEST(3),         ING(11)
       EQUIVALENCE   (FF,EF(1)),(DEPTH(1),SOURCE),(DEPTH(2),RECVRS(1)),
      1                         (RLOS(2),RLOSS(1))
       COMMON /AHZERO/ HZEROR,HZEROI
       DATA ( CONTR(I), I=1,4) /110.D0,95.D0,80.D0,-1000.D0/,
      1   (J SMBL(I), I=1,3) /1H1,1H*,1H8/,
      *(ING(I), I = 1,10)/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/,
      2   (TEST(I), I=1,3) /.2D0,1.D0,5.D0/
       TLIM = 25.
       SLIM = -8.54988
C   READ IN PARAMETERS
  1    READ 11, K1, K2, K3, K4, K5, K6, K7, K8, K9
C KEYS& 1-DEPT FN PRINTOUT, 2-LOSS PRINTOUT, 3-REFLECTION COEFF PRINTOUT
C   4-CHANGE CONTOURS, 5-CONTINUE MODES
 11    FORMAT (10I4)
       PRINT 13, K1, K2, K3, K4, K5, K6, K7, K8, K9
 13    FORMAT (6H1KEYS , 10I4)
       IF (K1 .LT. 7) GO TO 5
       READ 11, M
       READ 434, (SRES(I), I = 1,M)
 434   FORMAT (5D16.7)
  5    IF (K8 .NE. 1) GO TO 8
       READ 20, T LIM
       SLIM = -DCBRT(TLIM**2)
       PRINT 30, TLIM, SLIM
  8    EXPONT = DEXP((TLIM + TLIM) / 3.)
       K6 = K6 + 1
       INSUR = 0
       IF (K2 .LT. 10) GO TO 16
       K2 = K2 - 10
       INSUR = 1
```

```
      16  MPCH = 0
          IF (K5 .LT. 10) GO TO 17
          K5 = K5 - 10
          MPCH = 1
      17  IF (K4 .NE. 1) GO TO 3
          READ 20, (CONTR(I), I = 1,9)
          CONTR(10) = -1000.D0
          READ 4, (J SMBL(I), I = 1,9)
      4   FORMAT (9A1)
      3   READ 10, N, FREQ
      10  FORMAT (I2, D10.1)
          IF (N.EQ.0) GO TO 999
      2   PRINT 12, N, FREQ
      12  FORMAT (I3, 8H LAYERS, ,F10.1, 3H HZ)
          READ 20, (Z(I), I=1,N)
          PRINT30, (Z(I), I=1,N)
          READ 20, (C(I), I = 1,N)
      20  FORMAT (8D10.4)
          PRINT30, (C(I), I = 1,N)
          READ 20, (CB(I), I = 1,N)
          PRINT30, (CB(I), I = 1,N)
          READ 20, (GAMMA(I), I = 1,N)
          PRINT30, (GAMMA(I), I = 1,N)
          READ  20, (DPK(I), I = 1,N)
          PRINT 30, (DPK(I), I = 1,N)
          READ 20, (BLPK(I), I = 1,N)
          PRINT30, (BLPK(I), I = 1,N)
          READ  20, (RHO(I), I = 1,N)
          PRINT 30, (RHO(I), I = 1,N)
          IF (FREQ .GT. 0.) GO TO 18
          FREQ = - FREQ
          ATTEN = 0.
          GO TO 19
      18  F SQ = (FREQ / 1000.)**2
          ATTEN = .1 * F SQ / (1. + F SQ) + 40. * F SQ / (4100. + F SQ)
      19  ATTEN = ATTEN * 1.0936
          PRINT 14,ATTEN
      14  FORMAT (8H ATTEN = ,G10.5, 5HDB/KM )
          ATTEN = ATTEN / 1000.D0
      30  FORMAT (9F14.5)
    C  COMPLETE PROFILE
          DO 33 I = 1,N
          IF (RHO(I) .EQ. 0.) RHO(I) = 1.02
          IF (CB(I) .NE. 0.) GO TO 31
          CB(I) = C(I+1)
      31  IF (C(I) .NE. 0.) GO TO 32
          C(I) = CB(I-1)
      32  IF (DPK(I) .GE. 0.) GO TO 34
          CI(I) = CBI(I-1)
          GO TO 36
      34  CI(I) = 0.
          IF (DPK(I) .EQ. 0.) GO TO 36
          T = 27287.52708 / DPK(I)
          CI(I) = T - SQRT((T - C(I)) * (T + C(I)))
      36  CBI(I) = 0.
          IF (GAMMA(I) .NE. 0.) GO TO 38
    C     **BOTH SOUND SPEEDS GIVEN
```

```
      IF (BLPK(I) .LE. 0.) GO TO 37
      T = 27287.52708 / BLPK(I)
      CBI(I) = T - SQRT ((T - CB(I)) * (T + CB(I)))
   37 T = C(I) * (C(I)**2 - 3. * CI(I)**2)
      TI = CI(I) * (3. * C(I)**2 - CI(I)**2)
      IF (BLPK(I) .LT. 0.) GO TO 39
      TEMP = CB(I)**2 - CBI(I)**2
      TEMP1 = 2. * CB(I) * CBI(I)
      DENOM = TEMP**2 + TEMP1**2
      TEMP = TEMP / DENOM
      TEMP1 = -TEMP1 / DENOM
      GAMMA(I) = 0.5 * (C(I) - (T * TEMP - TI * TEMP1)) /
     * (Z(I+1) - Z(I))
      GAMMAI(I) = 0.5 * (CI(I) - (T * TEMP1 + TI * TEMP)) /
     * (Z(I+1) - Z(I))
      IF (I .EQ. N) GO TO 27
      GO TO 33
C     **SPECIAL CASE, GRADIENT REAL NUMBER
   39 IF (CI(I) .EQ. 0.) GO TO 42
      TEMP = CB(I)**2
      TEM = TEMP**2
      TEMP1 = CI(I)
      COEF1 = CI(I)
      COEF2 = 2. * TEMP * CI(I) + TI
      COEF3 = 2. * T * CB(I)
      COEF4 = TEM * CI(I) - TEMP * TI
      OLDFN = 1.D20
      DO 41 J = 1,10
      FN= (((COEF1 * TEMP1) + COEF2) * TEMP1 + COEF3) * TEMP1 + COEF4
      FP = ((4. * COEF1 * TEMP1) + 2. * COEF2) * TEMP1 + COEF3
      TEMP1 = TEMP1 - FN / FP
      IF (FN .GE. OLDFN) GO TO 43
      OLDFN = FN
   41 CONTINUE
   43 CBI(I) = TEMP1
      GAMMA(I) = .5 *(.5 * (CI(I) *(TEMP - CBI(I)**2) - TI) /
     * (CB(I) * TEMP1) + C(I)) / (Z(I+1) - Z(I))
      GO TO 28
   42 GAMMA(I) = C(I) - T / (CB(I)**2 * (Z(I+1) - Z(I)) * 2.)
      GO TO 33
C     **SOUND SPEED AND GRADIENT GIVEN
   38 IF (I .EQ. N) GO TO 33
      T = C(I) * (C(I)**2 - 3. * CI(I)**2)
      TI = CI(I) * (3. * C(I)**2 - CI(I)**2)
      IF (BLPK(I) .EQ. 0.) GO TO 29
      IF (BLPK(I) .LT. 0.) GO TO 28
      TEMP = (BLPK(I) / 54575.05416)**2 / (Z(I+1) - Z(I)) * 0.5
      T = T * TEMP
      TI = TI * TEMP
      TEMP = .5 * C(I) / (Z(I+1) - Z(I)) - GAMMA(I) + T
      T = -(TI - SQRT( TI * TI + T * TEMP)) / T
      CB(I) = 54575.05416 * T / BLPK(I) / (1. + T * T)
      CBI(I) = CB(I) / T
      GO TO 37
C     **SPECIAL CASE, GRADIENT REAL NUMBER
   28 TEMP = C(I) - 2. * GAMMA(I) * (Z(I+1) - Z(I))
      TEM = TEMP**2 + CI(I)**2
```

```
      XRE = (T * TEMP + TI * CI(I)) / TEM
      XIM = -(T * CI(I) - TI * TEMP) / TEM
      TEM = XRE**2 + XIM**2
      CB(I) = SQRT((XRE + SQRT(TEM)) * .5)
      CBI(I) = .5 * XIM / CB(I)
      GAMMAI(I) = 0.
      GO TO 33
 29   TEMP = C(I) - 2. * GAMMA(I) * (Z(I+1) - Z(I))
      CB(I) = SQRT (T / TEMP)
      GAMMAI(I) = .5 * (CI(I) - TI / CB(I)**2) / (Z(I+1) - Z(I))
      GO TO 33
 27   N = N - 1
 33   CONTINUE
C  COMPUTE USEFULL QUANTITIES
      PRINT 58
 58   FORMAT (7X,6H RE M ,8X,6H IM M ,9X,5H L/KM,8X,6H RE C ,8X,
     * 6H IM C ,5X,12H RE C BOTTOM,4X,12H IM C BOTTOM,10X,9H GRADIENT )
      OMEGA = 6.283185307D0 * FREQ
      DO 40 I = 1,N
      TEMP = C(I)**2 + CI(I)**2
      CAY(I) = OMEGA * C(I) / TEMP
      CAYI(I) = -OMEGA * CI(I) / TEMP
      CAY SQ(I) = CAY(I)**2 - CAYI(I)**2
      CAY SQI(I) = 2.D0 * CAY(I) * CAYI(I)
      TEMDR = -2. * (GAMMA(I) * CAY SQ(I) - GAMMAI(I) * CAY SQI(I))
      TEMDI = -2. * (GAMMA(I) * CAY SQI(I) + GAMMAI(I) * CAY SQ(I))
      G CU(I) = (TEMDR * C(I) + TEMDI * CI(I)) / TEMP
      G CUI(I) = (TEMDI * C(I) - TEMDR * CI(I)) / TEMP
      TEM1 = DCBRT(-DSQRT( GAMMA(I)**2 + GAMMAI(I)**2) * 2.*OMEGA**2)
      TEM1I = DATAN (ABS(GAMMAI(I) / GAMMA(I)))/ 3.
      CRTG = TEM1 * DCOS(TEM1I)
      CRTGI = TEM1 * DSIN(TEM1I)
      IF (GAMMA(I) .LT. 0.) CRTG = -CRTG
      IF (GAMMAI(I).LT. 0.) CRTGI = -CRTGI
      G(I) = (C(I) * CRTG + CI(I) * CRTGI) / TEMP
      GI(I) = (C(I) * CRTGI - CI(I) * CRTG) / TEMP
      CON(I) = G(I) * C(I) - GI(I) * CI(I)
      CON(I) = OMEGA**2 / CON(I)**2
      XMI = -GI(I) * (Z(I+1) - Z(I))
      XM = -G(I) * (Z(I+1) - Z(I))
      DPK(I) = -8686.D0 * CAYI(I)
      PRINT 30, XM, XMI, DPK(I), C(I), CI(I), CB(I), CBI(I)
     * ,GAMMA(I), GAMMAI(I)
      G SQI(I) = 2. * G(I) * GI(I)
 40   G SQ(I) = G(I)**2 - GI(I)**2
C  FIND MODES
      NXTRA=0
      IJ FLAG=0
      NN = NN + 1
      IF (K5 .EQ. 1) GO TO 15
      DO 50 NN = 1,350
 15   IF (IJ FLAG .EQ. 1) GO TO 53
 52   IF (NXTRA .GT. 0) GO TO 44
      READ 60, V,VI,STEP,STEPI,NXTRA
 60   FORMAT (4D10.4,I10)
      IF (NXTRA .GE.0) GO TO 62
      V = V + VI * 1.D-10
```

```
      VI = STEP + STEPI * 1.D-10
      GO TO 85
 62   IF (V) 142,301,70
 142  IF(STEP) 44,44,143
C  SEARCH FOR MODE
 143  SIZE3 = -1.
      SIZE2=0
      IJ FLAG=1
      V=-V
      IF (NXTRA) 55,55,54
 55   NXTRA = 20
 54   XTRA = NXTRA
      HOP = (STEP - V) / XTRA
      HOPI=0.
      IF(STEPI.NE.0.) HOPI=(STEPI-VI)/XTRA
      DO 47 IJ = 1,NXTRA
      CALL SETUP
      DET = VEL
      DETI = VELI
      CALL DETNT(N,VEL,VELI)
      DELTA = VEL
      DELTI = VELI
      SIZE = DELTA*DELTA + DELTI*DELTI
      PRINT 56, V, VI, SIZE, VEL, VELI
 56   FORMAT (2F12.3, 3D17.5)
      IF ((SIZE2.LT.SIZE3).AND.(SIZE.GT.SIZE2)) GO TO 45
   46 SIZE3=SIZE2
      SIZE2=SIZE
      V = V + HOP
      VI=VI+HOPI
      GO TO 47
 45   V = V - HOP
      TEMP = HOP / (SIZE - SIZE2)
      DELTI = TEMP * (DET * VELI - DETI * VEL)
      TEMP = .5D0 * (SIZE3 - SIZE) / (SIZE3 + SIZE - SIZE2 - SIZE2)
      DELTA = HOP * TEMP
      IF(HOPI.EQ.0) GO TO 49
      VI=VI-HOPI
      DELTAI=HOPI*TEMP
      GO TO 49
 47   CONTINUE
      IJ FLAG=0
      NXTRA=0
      GO TO 52
   53 SIZE3=-1.
      SIZE2=0
      GO TO 46
 44   NXTRA = NXTRA - 1
      V = 3. * (PHASE V(NN-1) - PHASE V(NN-2)) + PHASE V(NN-3)
      VI = 3.* (PHASI V(NN-1) - PHASI V(NN-2)) + PHASI V(NN-3)
      STEP = (PHASE V(NN-1) - PHASE V(NN-2)) * .0001
 70   CALL SETUP
      CALL DETNT(N,DET,DETI)
 80   FORMAT (/, 2D20.11, 4D13.4)
      VEL = DET
      VELI = DETI
      DELTA = STEP
```

```
      DELTI = STEPI
      IF (DELTA .NE.0.) GO TO 49
      IF (DELTI .EQ.0.) DELTA = .01
 49   SIZE2 = 100.
      RX = DET**2 + DETI**2
      IF (K6 .LT. 3) PRINT 80, V, VI, DET, DETI, SIZE, CNTR
      J = 0
 48   J = J + 1
      IF (J .GT. 15) GO TO 51
      V = V + DELTA
      VI = VI + DELTI
      IF (VI) 83,84,85
 83   DELTI = DELTI - VI
 84   VI = 1.D-18
 85   CALL SETUP
      NNN = N + N - 1
      DO 82 IA = 1,NNN
      DO 82 IB = 1,4
      BI(IA,IB) = Q(IA,IB)
 82   B(IA,IB) = A(IA,IB)
      CALL DETNT(N,DET,DETI)
      IF (K6 .NE. 1) GO TO 72
 71   PRINT 81, V, VI, DET, DETI, SIZE, CNTR
 81   FORMAT (2D20.11, 4D13.4)
 72   IF (NXTRA .LT. 0) GO TO 51
      TEMNR =  DET * DELTA - DETI * DELTI
      TEMNI = DETI * DELTA + DET * DELTI
      TEMDR = VEL - DET
      TEMDI = VELI - DETI
      TEMDEN = TEMDR*TEMDR + TEMDI*TEMDI
      IF (TEMDEN .EQ. 0.) GO TO 51
      TEMRNU = TEMNR*TEMDR + TEMNI*TEMDI
      TEMINU = TEMNI*TEMDR - TEMNR*TEMDI
      DELTA = TEMRNU/TEMDEN
      DELTI = TEMINU/TEMDEN
C * * * THE NEXT CONSTANT DEPENDS ON WORD LENGTH AND SIZE OF PHASE VELOCITY * *
      IF (ABS(DELTA) .LT. 1.D-14) GO TO 51
      SIZE = DELTA*DELTA + DELTI*DELTI
      IF ((SIZE.GT.SIZE2).AND.(J.GT.3)) GO TO 51
 92   SIZE2 = SIZE * 2.
      VEL = DET
      VELI = DETI
      GO TO 48
C  FIND DEPTH FUNCTIONS
 51   IF (INSUR .EQ. 0) GO TO 61
      TRE = (DET**2 + DETI**2) / RX
      IF (TRE .LT. 1E-10) GO TO 61
      PRINT 998, NN, TRE
 998  FORMAT (5H MODE ,I4,23H FAILED TO CONVERGE --  , E9.2)
      GO TO 999
 61   IF (MPCH .EQ. 0) GO TO 63
      IF (NXTRA .LT. 0) GO TO 63
      TEM1 = V * 1.D4
      COL(1) = TEM1
      TEMP = COL(1)
      COL(2) = (TEM1 - TEMP) * 1.D10
      TEM1 = VI * 1.D4
```

```
      COL(3) = TEM1
      TEMP = COL(3)
      COL(4) = (TEM1 - TEMP) * 1.D10
      COL(5) = -NN
      PUNCH 64, (COL(I), I = 1,5)
 64   FORMAT (5I10)
 63   AF(1,NN) = A(1,3)
      AG(1,NN) = Q(1,3)
      BF(1,NN) =-A(1,4)
      BG(1,NN) =-Q(1,4)
      PHASE V(NN) = V
      PHASI V(NN) = VI
      IF (K6 .EQ. 1) GO TO 73
 74   PRINT 81, V, VI, DET, DETI, SIZE, CNTR
 73   LL = N - 1
      IF (LL-1) 95,96,97
 96   I = 0
      GO TO 98
 97   DO 110 J = 2,LL
      I = J + J - 2
      TEMNR = A(I,2)*AF( J-1 ,NN) - Q(I,2)*AG( J-1 ,NN)
      TEMNI = Q(I,2)*AF( J-1 ,NN) + A(I,2)*AG( J-1 ,NN)
      TEMDR = A(I,3)*A( I+1 ,4) - Q(I,3)*Q( I+1 ,4) -
     1   A(I,4)*A( I+1 ,3) + Q(I,4)*Q( I+1 ,3)
      TEMDI = Q(I,3)*A( I+1 ,4) + A(I,3)*Q( I+1 ,4)  -
     1     Q(I,4)*A( I+1 ,3) - A(I,4)*Q( I+1 ,3)
      TEMDEN = TEMDR*TEMDR + TEMDI*TEMDI
      TEMRNU = TEMNR*TEMDR + TEMNI*TEMDI
      TEMINU = TEMNI*TEMDR - TEMNR*TEMDI
      TEMP = TEMRNU / TEMDEN
      TEMPI = TEMINU / TEMDEN
      BF(J,NN) = -(TEMP*A( I+1 ,4) - TEMPI*Q( I+1 ,4))
      BG(J,NN) = -(TEMPI*A( I+1 ,4) + TEMP*Q( I+1 ,4))
      AG(J,NN) = TEMPI*A( I+1 ,3) + TEMP*Q( I+1 ,3)
  110 AF(J,NN) = TEMP*A( I+1 ,3) - TEMPI*Q( I+1 ,3)
 98   TEMNR = -(A(I+2,2) * AF(LL,NN) -Q(I+2,2) * AG(LL,NN))
      TEMNI = -(Q( I+2 ,2)*AF(LL,NN) + A( I+2 ,2)*AG(LL,NN))
      TEMDEN = A( I+2 ,3)*A( I+2 ,3) + Q( I+2 ,3)*Q( I+2 ,3)
      TEMRNU = TEMNR*A( I+2 ,3) + TEMNI*Q( I+2 ,3)
      TEMINU = TEMNI*A( I+2 ,3) - TEMNR*Q( I+2 ,3)
      BF(N,NN) = TEMRNU / TEMDEN
      BG(N,NN) = TEMINU / TEMDEN
 95   AF(N,NN) = 0.
      AG(N,NN) = 0.
C  FIND NORMALIZING FACTOR
      D(NN) = 2.12429296D0 * RHO(1)**3 / G(1)
      DI(NN) = 0.
      DO 111 I = 2,N
      TEMRSP = AF( I-1 ,NN)*B( 2*I-2 ,2) - AG( I-1 ,NN)*BI( 2*I-2 ,2) +
     1   BF( I-1 ,NN)*B( 2*I-2 ,1) - BG( I-1 ,NN)*BI( 2*I-2 ,1)
      TEMISP = AG( I-1 ,NN)*B( 2*I-2 ,2) + AF( I-1 ,NN)*BI( 2*I-2 ,2)  +
     1   BG( I-1 ,NN)*B( 2*I-2 ,1) + BF( I-1 ,NN)*BI( 2*I-2 ,1)
      AX1 = TEMRSP*TEMRSP - TEMISP*TEMISP
      AX1I = TEMRSP * TEMISP
      AX1I = AX1I + AX1I
      TEMDR = (G(I-1)**2 + GI(I-1)**2)
      TEMDI = G(I)**2 + GI(I)**2
```

```
      TEMP = (RHO(I-1) / RHO(I)) / TEMDI
      TEM1  =(ZB (I-1) * G(I-1) + ZBI(I-1) * GI(I-1)) / TEMDR
     * -(ZT (I) * G(I) + ZTI(I) * GI(I)) * TEMP
      TEM1I =(ZBI(I-1) * G(I-1) - ZB (I-1) * GI(I-1)) / TEMDR
     * -(ZTI(I) * G(I) - ZT (I) * GI(I)) * TEMP
      TEMRSP = AF( I-1 ,NN)*B( 2*I-1 ,2) - AG( I-1 ,NN)*BI( 2*I-1 ,2)  +
     1   BF( I-1 ,NN)*B( 2*I-1 ,1) - BG( I-1 ,NN)*BI( 2*I-1 ,1)
      TEMISP = AG( I-1 ,NN)*B( 2*I-1 ,2) + AF( I-1 ,NN)*BI( 2*I-1 ,2)  +
     1   BG( I-1 ,NN)*B( 2*I-1 ,1) + BF( I-1 ,NN)*BI( 2*I-1 ,1)
      AX2 = TEMRSP*TEMRSP - TEMISP*TEMISP
      AX2I = TEMRSP * TEMISP
      AX2I = AX2I + AX2I
      TEMDR = RHO(I-1) / (G CU(I-1)**2 + G CUI(I-1)**2)
      TEMDI = RHO(I  ) / (G CU(I  )**2 + G CUI(I   )**2)
      TEM2 = G CU(I-1) * TEMDR - G CU(I) * TEMDI
      TEM2I = G CUI(I) * TEMDI - G CUI(I-1) * TEMDR
      TEMR1 = AX1*TEM1 - AX1I*TEM1I
      TEMI1 = AX1I*TEM1 + AX1*TEM1I
      TEMR2 = AX2 * TEM2 - AX2I * TEM2I
      TEMI2 = AX2I * TEM2 + AX2 * TEM2I
      D(NN) = D(NN) + TEMR1 / RHO(I-1) + TEMR2
      DI(NN) = DI(NN) + TEMI1 / RHO(I-1) + TEMI2
  111 CONTINUE
      IF ( K1 .GT. 3) DA(NN) = DSQRT((D(NN)**2 + DI(NN)**2) * FREQ /
     *  PHASE V(NN))
      EIGEN(NN) = LAMBDA
      EIGENI(NN) = LAMBDI
      IF (K6 .GT. 2) GO TO 131
      L = 0
      K = 24
      DO 112 I = 1,N
      L = L + 1
      COL(L) = SNGL(ZT(I)) * 100.
      L = L + 1
      COL(L) = SNGL(ZTI(I)) * 1000.
      K = K + 1
      COL(K) = SNGL(ZB(I)) * 100.
      K = K + 1
      COL(K) = SNGL(ZBI(I)) * 1000.
  112 CONTINUE
      PRINT 130, (COL(I), I=1,L)
      PRINT 130, (COL(I), I=25,K)
  130 FORMAT (4H Z = , 11(I6,I5))
      M = N + N
      PRINT 132, (RATIO(I), I = 1,M)
  132 FORMAT (11(1X,2F5.3))
  131 DB LOSS(NN) = - LAMBDI * 8686.D0
      PHINV = V * PHASE V(NN-1) /((V - PHASE V(NN-1))* FREQ)
      PRINT 109, NN,EIGEN(NN),EIGENI(NN),D(NN),DI(NN),PHINV,DB LOSS(NN)
  109 FORMAT (3H N=,I5,10H  LAMBDA =,2E15.7,4H D= ,2E15.7,
     * 12H INT RANGE = ,F8.0, 6H L/K =, F8.5)
      IF (K3 .EQ. 0) GO TO 50
      CALL RCOEF (K3)
   50 CONTINUE
C  READ IN SOURCE AND RECEIVERS DEPTHS
  301 NRT = NR
      NR = 0
```

```
      NN = NN - 1
      K1P1 = K1 + 1
      IF (K1 .NE. 3) GO TO 321
      NR = NRT
      GO TO 501
  321 READ 20, SOURCE
  320 NR = NR + 1
      READ 20, RECVRS(NR), FINAL, STEPP
      IF (NR.GT.50) GO TO 300
  350 IF (RECVRS(NR) .EQ.0.) GO TO 300
  310 IF (FINAL .EQ.0.) GO TO 320
  330 RECVRS(NR+1) = RECVRS(NR) + STEPP
      IF (RECVRS(NR+1) .GT. FINAL) GO TO 320
  340 NR = NR + 1
      IF(NR .GT. 50) GO TO 300
      GO TO 330
  300 PRINT 303
  303 FORMAT (/21H SOURCE AND RECEIVERS )
      PRINT 21,(DEPTH(I), I = 1,NR)
   21 FORMAT (8F10.2)
C  COMPUTE DEPTH FUNCTIONS
      DO 500 I = 1,NN
      LOC = 1
      DO 305 J = 1,NR
      IF ((J .EQ. 1) .AND. (K1 .GT. 5)) GO TO 305
      LCTR = 0
  380 IF((DEPTH(J) .GE. Z(LOC)).AND.(DEPTH(J) .LT. Z(LOC+1)))GO TO 360
  371 IF (LOC .GE. N) GO TO 385
  370 LOC = LOC + 1
      GO TO 380
  385 IF (DEPTH(J) .GE. Z(LOC)) GO TO 360
   390 LOC=1
      LCTR=LCTR+1
      IF (LCTR .GT. 2) GO TO 305
      GO TO 380
  360 X1 = CAY (LOC) - EIGEN (I)
      X2 = CAY (LOC) + EIGEN (I)
      X3 = CAYI(LOC) - EIGENI(I)
      X4 = CAYI(LOC) + EIGENI(I)
      TEMP = X1 * X2 - X3 * X4
      TEMPI = X1 * X4 + X3 * X2
      TEMDEN = G SQ(LOC) **2 + G SQI(LOC)**2
      ZE = (TEMP * GSQ(LOC) + TEMPI * G SQI(LOC)) / TEMDEN
      ZEI = (TEMPI * GSQ(LOC) + TEMP * GSQI(LOC)) / TEMDEN
      TEM1 = ZE
      IF (ZE .GT. -7.5) GO TO 438
      S = CAY(LOC)
      T = CAYI(LOC)
      DO 437 K = 1,20
      TEMP = S**2 + T**2
      TEMPI = (EIGENI(I) * S - EIGEN(I) * T) / TEMP
      TEMP = (EIGEN(I) * S + EIGENI(I) * T) / TEMP
      ZE = ((1. + TEMP) * (1. - TEMP) + TEMPI**2) * CON(LOC)
      ZEI = -2. * TEMPI * TEMP * CON(LOC)
      ZR = ZE / -7.5
      IF (DABS(ZR-1.) .LT. 1.D-3) GO TO 438
      S = EIGEN(I) + (S - EIGEN(I)) / ZR
```

```
437  CONTINUE
438  IF (G(LOC) .LT. 0.) GO TO 439
     ZE = G(LOC) * (DEPTH(J) - Z(LOC)) + TEM1
     IF (ZE .GT. -7.5) GO TO 442
     F(J) = 1.D-12
     FI(J) = 0.
     GO TO 305
439  ZE = G(LOC) * (DEPTH(J) - Z(LOC)) + ZE
     IF (ZE .GT. SLIM) GO TO 442
     F(J) = 1.D-12
     FI(J) = 0.
     GO TO 305
442  ZEI =GI(LOC) * (DEPTH(J) - Z(LOC)) + ZEI
302  CALL HANKEL(ZE,ZEI,1)
     F(J) =(AF(LOC,I)*H1R - AG(LOC,I)*H1I + BF(LOC,I)*H2R - BG(LOC,I)
    1    *H2I) * RHO(LOC)
     FI(J) =(AG(LOC,I)*H1R + AF(LOC,I)*H1I + BG(LOC,I)*H2R + BF(LOC,I)
    1    *H2I) * RHO(LOC)
305  CONTINUE
     IF (K1 .EQ. 2) GO TO 451
     GO TO 432
451  PRINT 270, DEPTH(NR)
270  FORMAT(7H1DEPTH ,F5.1,6X,3HE-8,17X,3HE-6,17X,3HE-4,17X,3HE-2,
    *       17X,3HE 0    )
432  IF (K1 .LT. 4) GO TO 431
     IF (K1 .GT. 5) GO TO 433
     SRES(I) = (F(1)**2 + FI(1)**2) / DA(I)
     GO TO 500
 431 TEMDEN = D(I)*D(I) + DI(I)*DI(I)
     TEMRE = F(1)*D(I) + FI(1)*DI(I)
     FD = TEMRE/TEMDEN
     FDI = (D(I) * FI(1) - DI(I) * F(1)) / TEMDEN
433  DO 400 K = 2,NR
     J = K - 1
     L = J * NN - NN + I
     IF (K1 .LT. 6) GO TO 435
     FF = SRES(I) * (F(K)**2 + FI(K)**2) / DA(I)
     GO TO 436
435  FF = FD * F (K) - FDI * FI(K)
     FFI = FD * FI(K) + FDI * F(K)
436  UU(L) = FF
     UUI(L) = FFI
452  GO TO (400,410,420,400,400,400,400,400), K1P1
C  PLOT DEPTH FUNCTIONS
420  DO 210 II = 1,120
210  COL(II)= 1H
     DO 220 II= 20,100,20
220  COL(II)= 1HI
     FE = FF * FF + FFI * FFI
     IF ((FE.GT.1E-20).AND.(FE.LT.10000.)) GO TO 240
     GO TO 250
240  INT = 100.D0 + 2.17147D0 * DLOG(FE)
     COL(INT) = 1H*
     GO TO 225
250  COL(2) = 1H*
225  PRINT 260, COL
260  FORMAT (120A1)
```

```
      GO TO 400
C  PRINT DEPTH FUNCTIONS
  410 F MAG(J) = SQRT (FF * FF + FFI * FFI)
      IF (FF) 430,440,450
  430 F ANG(J) = ATAN(FFI / FF) * 57.29577951D0 + 180.D0
      GO TO 400
  440 F ANG(J) = 90.
      GO TO 400
  450 F ANG(J) = ATAN(FFI / FF) * 57.29577951D0
  170 FORMAT (  10F12.4)
  180 FORMAT(/10E12.3)
  400 CONTINUE
      IF (K1.EQ.1) GO TO 441
      GO TO 500
  441 PRINT 180, (F MAG(K), K = 1,J)
      PRINT 170, (F ANG(K), K = 1,J)
  500 CONTINUE
C  CALCULATE ATTENUATION AND READ IN RANGES
      IF ((K1 .EQ. 4).OR.(K1 .EQ. 5)) PRINT 180, (SRES(K), K= 1,NN)
      IF (K1 .EQ. 5) PUNCH 434, (SRES(K), K = 1,NN)
      NR=NR-1
      IF (K2 .LT. 3) GO TO 501
      IF (K2 .EQ. 4) K8 = 3
      IF (K2 .EQ. 3) K8 = 2
      K2 = 0
  501 KX = K2 + 1
      GO TO (561,551,551), KX
  551 PRINT 533, NN,N, C(1), Z(2), C(2), Z(3), C(3), Z(4), C(4),
     *  SOURCE, RECVRS(40), FREQ
  533 FORMAT (1H1, 2I5, 10F10.4)
      ICTR=0
      R LOS(1) = 120.
      LEVEL(1) = 1
      DO 562 I = 1,5
      P LEV(I)=40.
      J COU(I)=4
      J COUNT(I)=-6
  562 CONTINUE
      IF((K2 .EQ. 2).AND.(NR .GT. 5))GO TO 772
      GO TO 561
  772 NR = 5
  561 NL = NN
      PRINT 524, NL
  524 FORMAT (I8, 13H MODES IN SUM  )
      LL = 1
      IF (K9 .GT. 0) NL = K9
      READ 20, RANGE, FINAL R, STEP R
      IF (K8 .EQ. 3) PUNCH 30, RANGE, FINAL R, STEP R
      IF (RANGE) 563,1,564
  563 NN = NN + 1
      READ 11, K1, K2, K3, K4, K5, K6, K7, K8, K9
      PRINT 11, K1, K2, K3, K4, K5, K6, K7, K8, K9
      GO TO 301
  564 IF (FINALR .LE. 0.) GO TO 550
      FINAL R = FINAL R + 1.D-3
  560 IF (RANGE .GE. FINALR) GO TO 561
  550 R ATTEN = RANGE * ATTEN - 9.9429946
```

```
      IF (K1 .GT. 5) RATTEN = 0.D0
      IF (K1 .GT. 5) GO TO 536
      IF (K7 .EQ. 2) RATTEN = 4.3429448D0 * DLOG(FREQ)
      IF (RANGE * DB LOSS(NL) .LT. 15.D4) GO TO 522
  521 NL = NL - 1
  522 DO 520 I = 1,NL
      IF (K7 .LT. 2) GO TO 523
      FMAG(I) = PHASE V(I)
      GO TO 520
  523 TEMP RE = EIGEN(I) * RANGE
      TEMPIM = EIGENI(I)*RANGE
      CALL HZERO(TEMPRE,TEMPIM)
      HZERO2(I) = HZEROR
      IF (K7 .EQ. 0) GO TO 520
      FMAG(I) = HZEROR**2 + HZEROI**2
  520 HZER2I(I) = HZEROI
  536 L = 0
      DO 540 J = 1,NR
      FF = 0
      FFI = 0
      TEMP = 0.D0
      DO 530 I = 1,NL
      K = L + I
      IF (K1 .LT. 6) GO TO 537
      TEMP = TEMP + UU(K)
      GO TO 530
  537 IF (K7 .EQ. 0) GO TO 534
      TEMP = TEMP + FMAG(I) * (UUI(K)**2 + UU(K)**2)
      GO TO 530
  534 TEMIM = UUI(K) * HZERO2(I) + UU(K) * HZER2I(I)
      TEMRE = UU(K) * HZERO2(I) - UUI(K) * HZER2I(I)
      FF = FF + TEMRE
      FFI = FFI + TEMIM
  530 CONTINUE
      IF (K1 .GT. 5) GO TO 535
      IF (K7 .GT. 0) GO TO 535
      TEMP = FF**2 + FFI**2
  535 T RE = TEMP
      RX = -4.3429448 * ALOG(T RE) + R ATTEN
      R LOSS(J) = RX
      IF (K4 .LT. 2) GO TO 545
      T RE = -4.3429448 * ALOG(UU(K)**2 + UUI(K)**2)
      PRINT 170, RECVRS(J), RX, T RE
      IF (K4 .NE. 3) GO TO 545
  545 CONTINUE
      L = L + NN
      IF (K8 .LT. 2) GO TO 540
      IF (K8 .EQ. 3) GO TO 538
      LPCH = -RLOSS(J) * 10.D0 + 1400.5D0
      IF (LPCH .LT. 0) LPCH = 0
      IF (LPCH .GT. 999) LPCH = 999
      LOSPCH(J,LL) = LPCH
      IRNG = RANGE / 1000.D0
      IF (LL .EQ. 25) PUNCH 903, IRNG, (LOSPCH(J,LLL),LLL = 1,25)
  903 FORMAT (I5,25I3)
      GO TO 540
  538 CONTINUE
```

```
            LOSS(J) = (140.05 - RX) * 10.
            IF (LOSS(J) .LT. 0) LOSS(J) = 0
            IF (LOSS(J) .GT. 999) LOSS(J) = 999
      540   CONTINUE
            GO TO (770,780,716),KX
C  PLOT DB LOSSES
      712   COL(15)=1HI
            COL(39)=1HI
            COL(63)=1HI
            COL(87)=1HI
            COL(111)=1HI
            COL(27)=1HX
            COL(51)=1HX
            COL(75)=1HX
            COL(99)=1HX
            I PLACE = 135
            DO 787  J = 1,NR
            I PLACE = I PLACE - 24
      783   IPLOT = P LEV(J) -   R LOSS(J)
            IF (I PLOT .GT. 10) GO TO 776
            GO TO 777
      776   P LEV(J) = P LEV(J) - 20.
            J COUNT(J) = J COUNT(J) - 2
            J COU(J)=J COU(J)-2
      786   COL(I PLACE + 1) = 1HO
            IF (P LEV(J) - 100.) 778,779,781
      778   JC = J COU(J) + 1
            COL(IPLACE) = ING(JC)
            GO TO 783
      779   COL(I PLACE) = 1HO
            GO TO 782
      781   JC = J COUNT(J) + 1
            COL(IPLACE) = ING(JC)
      782   COL(I PLACE - 1) = 1H1
            GO TO 783
      777   IF (I PLOT  .LT. -9) GO TO 784
            GO TO 785
      784   P LEV(J) = P LEV(J) + 20.
            J COU(J)=J COU(J)+2
            J COUNT(J) = J COUNT(J) + 2
            GO TO 786
      785   IPP = I PLACE + IPLOT
            COL(IPP) = 1H+
      787   CONTINUE
            GO TO 750
C  CONTOUR LOSS FIELD
      780   DO 590 JJ = 1,120
      590   COL(JJ) = 1H
            COL(31)=1HI
            COL(61)=1HI
            COL(91)=1HI
            DO 640 JJ = 2,41
            LEV = 1
      620   IF (RLOS (JJ) .LT. CONTR(LEV)) GO TO 600
            GO TO 610
      600   LEV=LEV+1
            GO TO 620
```

```
610  IF (LEV .EQ. LEVEL(JJ)) GO TO 840
650  IF (LEV .GT. LEVEL(JJ)) GO TO 660
     GO TO 670
660  II = LEVEL(JJ)
     GO TO 680
670  II = LEV
680  JJJ = 124 - 3*JJ
     COL(JJJ) = J SMBL(II)
     LEVEL(JJ) = LEV
640  CONTINUE
     COL(1) = 1HI
      PRINT 261, (COL(I1), I1 = 1,119)
716  DO 690 JJ = 1,120
690  COL(JJ) = 1H
     ICTR = ICTR + 1
     IF (ICTR .EQ. 10) GO TO 700
     GO TO 714
700  TEMP = (RANGE + 1.) / 10000.
     IND = TEMP
     COL(2) = ING(IND+1)
     TEMP1 = IND
     TEMP = (TEMP - TEMP1) * 10.
     IND = TEMP
     COL(3) = ING(IND+1)
     TEMP1 = IND
     IND = (TEMP - TEMP1) * 10.
     COL(5) = ING(IND+1)
     COL(4)=1H.
     COL(6)=1HK
     COL(7)=1HY
     COL(8)=1HD
     COL(9) = 1HS
     ICTR=0
714  GO TO (710,712),K2
 710 COL(31)=1HI
     COL(61)=1HI
     COL(91)=1HI
     DO 720 JJ = 1,40
     TEMP = LEVEL(JJ)
     TEMPI = 0.
830  IF (LEVEL(JJ) .GT. LEVEL(JJ+1)) GO TO 730
     GO TO 740
730  II = LEVEL(JJ) - 1
     KK = 1
860  EX = (CONTR(II) - R LOS (JJ) )/ (RLOS (JJ+1) - CONTR(II))
     DO 760 LL = 1,3
     IF (EX .LT. TEST(LL)) GO TO 800
760  CONTINUE
     LL = 4
800  JJLL = 125 - 3*JJ - LL
     COL(JJLL) = J SMBL(II)
     GO TO (810,820),KK
810  LEVEL(JJ) = LEVEL(JJ) - 1
     GO TO 830
740  IF (LEVEL(JJ) .LT. LEVEL(JJ+1)) GO TO 840
     GO TO 720
840  II = LEVEL(JJ)
```

```
          KK=2
          GO TO 860
     820  LEVEL(JJ) = LEVEL(JJ) + 1
          GO TO 740
     720  LEVEL(JJ) = TEMP
          COL(1) = 1HI
     750   PRINT 261, (COL(I1), I1 = 1,119)
     261  FORMAT (1X, 119A1)
          GO TO 581
C  PRINT DB LOSSES
     770  PRINT 580, RANGE, (R LOSS(K), K = 1,NR)
          LL = LL + 1
          IF (LL .GT. 25) LL = 1
     580  FORMAT (F9.0, 2X, 18F6.1)
     581  RANGE = RANGE + STEP R
          IF (K8 .NE. 3) GO TO 560
          PUNCH 980, (LOSS(I), I= 1,NR)
     980  FORMAT (26I3)
          GO TO 560
     999  STOP
          END
```

```
      SUBROUTINE SETUP
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION  LAMBDA,LAMBDI
      COMMON /HAN/ H2R,H2I,H1R,H1I,H2PR,H2PI,H1PR,H1PI,EXPONT
      COMMON /EXPO/ EXSUM, CNTR, RATIO(25)
      COMMON/DETMNT/ A(25,4),Q(25,4)
      COMMON/INPUT/ Z(12), N, OMEGA, V, VI, CON(12), GSQ(12),
     1 CAY(12), LAMBDA, LAMBDI, G(12)
     2,RHO(12), GI(12), G SQI(12), CAYI(12)
       COMMON /LIMIT/ TLIM, EXPON , SLIM
      COMMON/PARTS/ ZT(12),ZTI(12),ZB(12),ZBI(12)
      DENOM = V * V + VI * VI
      LAMBDA = OMEGA * V / DENOM
      LAMBDI = -OMEGA * VI/ DENOM
      M = N - 1
      DO 10 I = 0,M
      IF (I .EQ. 0) GO TO 35
      IF (ZR .GT. -7.4) GO TO 25
      IF (G(I) .LT. 0.) GO TO 25
      ZE = G(I) * (Z(I+1) - Z(I)) + ZE
      IF (ZE .LT. -7.5) ZE = -7.5
      GO TO 26
   25 CONTINUE
      ZE = G(I) * (Z(I+1) - Z(I)) + ZR
      IF (ZE .LT. SLIM) ZE = SLIM
   26 CONTINUE
      ZQ = GI(I) * (Z(I+1) - Z(I)) + ZI
   30 ZB(I) = ZE
      ZBI(I) = ZQ
      CALL HANKEL(ZE,ZQ,0)
      ZB(I) = ZE
      ZBI(I) = ZQ
      RATIO(2*I) = EXPONT
      A(2*I,1) = H2R * RHO(I)
      Q(2*I,1) = H2I * RHO(I)
      A(2*I,2) = H1R * RHO(I)
      Q(2*I,2) = H1I * RHO(I)
      A(2*I+1,1) = H2PR * G(I) - H2PI * GI(I)
      Q(2*I+1,1) = H2PI * G(I) + H2PR * GI(I)
      A(2*I+1,2) = H1PR * G(I) - H1PI * GI(I)
      Q(2*I+1,2) = H1PI * G(I) + H1PR * GI(I)
   35 CONTINUE
      GSABS = G SQ(I+1)**2 + G SQI(I+1)**2
      X1 = CAY(I+1) - LAMBDA
      X2 = CAY(I+1) + LAMBDA
      X3 =CAYI(I+1) - LAMBDI
      X4 =CAYI(I+1) + LAMBDI
      X = X1 * X2 - X3 * X4
      Y = X1 * X4 + X3 * X2
      ZT(I+1) = (X * G SQ(I+1) + Y * G SQI(I+1)) / GSABS
      ZTI(I+1) = (Y * G SQ(I+1) - X * G SQI(I+1)) / GSABS
      ZR = ZT(I+1)
      ZI = ZTI(I+1)

      ZE = ZR
      ZQ = ZI
```

```
      IF (ZR .GT. -7.5) GO TO 40
      S = CAY(I+1)
      T = CAYI(I+1)
      CON =           (G(I+1) * S + GI(I+1) * T) / (S**2 + T**2)
      CON = 1. / CON**2
      DO 36 J = 1,20
      TEMP = S**2 + T**2
      TEMPI = (LAMBDI * S - LAMBDA * T) / TEMP
      TEMP = (LAMBDA * S + LAMBDI * T) / TEMP
      ZR = ((1. + TEMP) * ( 1. - TEMP) + TEMPI**2) * CON(I+1)
      R = ZR / -7.5
      IF (DABS(R-1.) .LT. 1.D-3) GO TO 41
      S = LAMBDA + (S - LAMBDA) / R
   36 CONTINUE
   41 ZI = -2. * TEMPI * TEMP * CON(I+1)
      ZT(I+1) = ZR
      ZTI(I+1) = ZI

   40 CONTINUE
      CALL HANKEL(ZR,ZI,0)
      ZT(I+1) = ZR
      ZTI(I+1) = ZI
      RATIO(2*I+1) = EXPONT
      IF (I .NE. 0) GO TO 45
      A(1,3) = H2R * RHO(1)
      Q(1,3) = H2I * RHO(1)
      A(1,4) = H1R * RHO(1)
      Q(1,4) = H1I * RHO(1)
      GO TO 10
   45 CONTINUE
      A(2*I,3) =-H2R * RHO(I+1)
      Q(2*I,3) =-H2I * RHO(I+1)
      A(2*I,4) =-H1R * RHO(I+1)
      Q(2*I,4) =-H1I * RHO(I+1)
      A(2*I+1,3) =-H2PR * G(I+1) + H2PI * GI(I+1)
      Q(2*I+1,3) =-H2PI * G(I+1) - H2PR * GI(I+1)
      A(2*I+1,4) =-H1PR * G(I+1) + H1PI * GI(I+1)
      Q(2*I+1,4) =-H1PI * G(I+1) - H1PR * GI(I+1)
   10 CONTINUE
      A( 2*N-2 ,4) = 0.
      Q( 2*N-2 ,4) = 0.
      A( 2*N-1 ,4) = 0.
      RETURN
      END
```

```
      SUBROUTINE DETNT(N,DET,DETI)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION DET, DETI
      COMMON /EXPO/ EXSUM, CNTR, RATIO(25)
      COMMON/DETMNT/ A(25,4), Q(25,4)
      DLOSS = 1.
      CNTR = 0.
      DET = A(1,3)
      DETI = Q(1,3)
      LIM = N + N - 3
      DO 100 I=1,LIM,2
      J = I
      K = 4
      L = I
      M = 3
      II = 1
      GO TO 500
   10 J = I + 1
      K = 2
      L = J
      M = 1
      GO TO 600
   30 J = I + 2
      L = J
      II = 2
      GO TO 600
   40 L = I + 1
      M = 2
      GO TO 500
   50 K = 3
      M = 3
      II = 3
      GO TO 600
   60 K = 4
      IF (I .EQ. LIM) GO TO 70
      M = 4
      II = 4
      GO TO 600
   70 K = 3
      II = 2
      GO TO 700
  500 C = A(L,M)*A(L,M) + Q(L,M)*Q(L,M)
 80   B = (A(J,K)*A(L,M) + Q(J,K)*Q(L,M)) / C
      BI = (Q(J,K)*A(L,M) - A(J,K)*Q(L,M))/C
      GO TO (10,50), II
  600 TD = A(J,K) - (A(L,M)*B - Q(L,M)*BI)
      TDI = Q(J,K) - (A(L,M)*BI + Q(L,M)*B)
      TEM = TD**2 + TDI**2
      TEMP = A(J,K)**2 + Q(J,K)**2
      TEMP = TEM / TEMP
      IF (II .EQ. 2) GO TO 92
      IF (II .EQ. 4) GO TO 92
      Q(J,K) = Q(J,K) * 10.D-18
      A(J,K) = A(J,K) * 10.D-18
      IF (TEMP .GT. 10.D-35) GO TO 92
      CNTR = CNTR + 1.
```

```
      GO TO 90
92    A(J,K) = TD
      Q(J,K) = TDI
90    GO TO (700,40,60,70),II
 700  C    = DET*A(J,K) - DETI*Q(J,K)
      DETI = DET*Q(J,K) + DETI*A(J,K)
      DET = C
      GO TO (30,100), II
100   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RCOEF (K3)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/INPUT/ Z(12), N, OMEGA,     V, VI, GCU(12), GSQ(12),
     1 CAY(12), LAMBDA, LAMBDI, G(12)
     2,RHO(12), GI(12), G SQI(12), CAYI(12)
      DIMENSION RR(12), RI(12), RA(12), RT(12), CYSQ(12), CYSQI(12)
      COMMON/REFL/ AF(12,200), AG(12,200), BF(12,200), BG(12,200),
     2 EIGEN(350), EIGENI(350),BR(25,4), BI(25,4), CB(12), CBI(12),
     3 CAYSQ(12), CAYSQI(12), NN
      NM = N - 1
      I = K3
      IF (I .GT. NM) I = NM
 110  J = I + I
      K = J + 1
      IF (NN .NE. 1) GO TO 102
      L = I + 1
      TEMP = CB(I)**2 + CBI(I)**2
      CY = OMEGA * CB(I) / TEMP
      CYI = -OMEGA * CBI(I) / TEMP
      CYSQ(I) = CY**2 - CYI**2
      CYSQI(I) = CY * CYI
      CYSQI(I) = CYSQI(I) + CYSQI(I)
 102  EL SQ = C YSQ(I) - EIGEN(NN)**2 + EIGENI(NN)**2
      ELSQI = C YSQI(I) - 2.D0 * EIGEN(NN) * EIGENI(NN)
      TEMP = ELSQ + DSQRT (ELSQ**2 + ELSQI**2)
      IF (TEMP .LE. 0.D0) GO TO 107
      EL = DSQRT (TEMP * .5D0)
      ELI = ELSQI / (EL + EL)
 103  A = AF(I,NN)*BR(J,2) - AG(I,NN)*BI(J,2)
     * + BF(I,NN)*BR(J,1) - BG(I,NN)*BI(J,1)
      B = AF(I,NN)*BI(J,2) + AG(I,NN)*BR(J,2)
     * + BF(I,NN)*BI(J,1) + BG(I,NN)*BR(J,1)
      E = AF(I,NN)*BR(K,2) - AG(I,NN)*BI(K,2)
     * + BF(I,NN)*BR(K,1) - BG(I,NN)*BI(K,1)
      F = AF(I,NN)*BI(K,2) + AG(I,NN)*BR(K,2)
     * + BF(I,NN)*BI(K,1) + BG(I,NN)*BR(K,1)
      C = (F * EL - E * ELI) / (ELSQ + ELSQI)
      D = -(E * EL + F * ELI) / (ELSQ + ELSQI)
      TEMP =  (A + C)**2 + (B + D)**2
      RR(I) = (A**2 - C**2 + B**2 - D**2) / TEMP
      RI(I) = -2.D0 * (A * D - B * C) / TEMP
 10   FORMAT (10D13.5)
      RA(I) = 0
      IF (CB(I) .GT. V) GO TO 104
      RX = CB(I) / V
      RA(I) = ACOS(RX) * 57.296
 104  RT(I) = RR(I)**2 + RI(I)**2
      RT(I) = 1.D0 / RT(I)
      RI(I) = -DATAN2 (RI(I), RR(I)) * 57.296D0
      RR(I) = -4.34294D0 * DLOG (RT(I))
      IF (K3 .NE.1) GO TO 108
      I = I + 1
      IF (I .LT. N) GO TO 110
 105  CONTINUE
      PRINT 106, (RR(I), I = 1,NM)
      PRINT 106, (RI(I), I = 1,NM)
```

```
          PRINT 106, (RT(I), I = 1,NM)
          PRINT 106, (RA(I), I = 1,NM)
    106   FORMAT (9G13.4)
          RETURN
    108   PRINT 109, Z(L), RR(I), RT(I), RI(I), RA(I)
    109   FORMAT (9H AT DEPTH, F7.0,8H YD, R = ,F9.4 ,7H DB, OR,D12.4,
         * 8H, PH A = ,F9.3,16H DEGREES, GR A =,F8.2,9H DEGREES. )
          RETURN
    107   EL = 0.D0
          ELI = DSQRT ( DABS (ELSQ))
          GO TO 103
          END
```

```
      SUBROUTINE HANKEL (ZR, ZI, IH)
      COMMON /HAN/ H2R,H2I,H1R,H1I,H2PR,H2PI,H1PR,H1PI,R
      INTEGER FLPS, FLQUAD
      DOUBLE PRECISION ZMLA2, TLIM, R
      COMMON /LIMIT/ TLIM,EXPONT
      DOUBLE PRECISION    ZR,ZI,H2R,H2I,H1R,H1I,H2PR,H2PI,H1PR,H1PI,A0,
     1  A,B0,B,C0,C,D0,D,     C4,     D4,K,K1,K2,CON4,STORE1,STORE2,
     2  STORE3,STORE4,STORE5,STORE6,STORE7,STORE8,STORE9,STOR10,STOR11,
     3  STOR12,STOR13,STOR14,STOR15,STOR16,STOR18,C1,C2,C3,CPR,CPI,CTHR,
     4  CTP,FR,FI,FPR,FPI,              F1R,F1I,F2R,F2I,GR,GI,GPR,GPI,G1R,
     5  G1I,G2R,G2I,H11R,H11I,H12R,H12I,H11PR,H11PI,H12PR,H12PI,
     6  H21PR,H21PI,H22PR,H22PI,H21R,H21I,H22R,H22I,       PI,SR,SI,SPR,
     7  SPI,S2,     THR,X,XR,XI,XPR,XPI,YR,YI,              ZM,ZMSQ,ZM1R,ZM1I,
     8  EXPONT,ABK, ZRTM2R,ZRTM2I,ZRTM4R,ZRTM4I,ZRT2R,ZRT2I,ZRT2M,ZRT4R,
     9  ZRT4I,       Z32R,Z32I,STP,STOR17,STHR,C5,D5,FPI12
      DIMENSION A(40),B(40),C(40),D(40),C4(20),D4(20),ZMLA2(40),
     1 XPR(40), XPI(40), C5(20), D5(20), ZMLA5(20)
      DATA (A(I),I=1,36) /
     *              -1.5507278615487157D-001,     5.1690928718290523 7D-03,
     1   -7.1792965531812830D-05,     5.4388603449379759 6D-07,
     2   -2.5899334975895123 7D-09,    8.4638349594428508 9D-12,
     3   -2.0151987998673454 5D-14,    3.6507224635277997 3D-17,
     4   -5.2004593497547004 7D-20,    5.9775394824766672 0D-23,
     5   -5.6605487523453287 9D-26,    4.4924990097978799 9D-29,
     6   -3.0313758500660458 8D-32,    1.7603808653112926 1D-35,
     7   -8.8908124510671344 1D-39,    3.9409629658985524 9D-42,
     8   -1.5454756729013931 3D-45,    5.3999848808574183 5D-49,
     9   -1.6917245867347801 8D-52,    4.7788830133750852 7D-56,
     A   -1.2234723536546557 3D-59,    2.8519169082859107 8D-63,
     B   -6.0782542802342514 6D-67,    1.1890168779800961 4D-70,
     C   -2.1423727531172903 4D-74,    3.5670542009944894 1D-78,
     D   -5.5047132731396441 5D-82,    7.8954579362301264 3D-86,
     E   -1.0552603496698912 6D-89,    1.3174286512732724 9D-93,
     F   -1.5397716821800753 7D-97,    1.6883461427413107 2D-101,
     G   -1.7402042287583083 0D-105,   1.6891906705089383 6D-109,
     H   -1.5468779033964637 0D-113,   1.3385928551371267 8D-117/

      DATA (B(I),I=1,36) /
     *              -5.6524893762022989D-002,     1.3458308038576902 2D-03,
     1   -1.4953675598418780 2D-05,    9.5856894861658847 6D-08,
     2   -3.9940372859024519 9D-10,    1.1678471596206000 0D-12,
     3   -2.5278077048064935 0D-15,    4.2130128413441558 3D-18,
     4   -5.5727683086562907 8D-21,    5.9922239878024632 0D-24,
     5   -5.3406630907330331 6D-27,    4.0095068248746495 2D-30,
     6   -2.5701966826119548 2D-33,    1.4231432351118243 7D-36,
     7   -6.8750880923276539 8D-40,    2.9230816719080161 5D-43,
     8   -1.1022178250030226 8D-46,    3.7111711279563053 2D-50,
     9   -1.1225563000472792 9D-53,    3.0670937159761729 1D-57,
     A   -7.6068792558932860 0D-61,    1.7202350194240809 6D-64,
     B   -3.5615631872134181 3D-68,    6.7761856682142658 5D-72,
     C   -1.1888045031954852 4D-75,    1.9292510600381130 1D-79,
     D   -2.9046236977388030 9D-83,    4.0681004170011247 7D-87,
     E   -5.3136107850066938 0D-91,    6.4879252564184295 5D-95,
     F   -7.4215571452967622 4D-99,    7.9698852505334646 0D-103,
     G   -8.0503891419529945 5D-107,   7.6626586159841943 1D-111,
     H   -6.8846887834539032 5D-115,   5.8483594830563228 4D-119/
```

```
      DATA (C(I),I=1,36) /
     *          -3.1014557230974314D-002,    6.4613660897863154 7D-04,
     1    -6.5266324139255711 8D-06,    3.8849002463842685 6D-08,
     2    -1.5234902926997131 6D-10,    4.2319174797214254 4D-13,
     3    -8.7617339124667193 5D-16,    1.4041240244337691 3D-18,
     4    -1.7932618447430001 6D-21,    1.8679810882739585 0D-24,
     5    -1.6172996435272368 0D-27,    1.1822365815257578 9D-30,
     6    -7.3935996343074289 7D-34,    4.0008656029802104 8D-37,
     7    -1.8916622236313051 9D-40,    7.8819259317971049 7D-44,
     8    -2.9159918356630059 1D-47,    9.6428301443882470 5D-51,
     9    -2.8673298080250511 6D-54,    7.7078758280243310 8D-58,
     A    -1.8822651594687011 2D-61,    4.1939954533616335 1D-65,
     B    -8.5609215214566922 0D-69,    1.6067795648379677 5D-72,
     C    -2.7823022767757017 4D-76,    4.4588177512431117 6D-80,
     D    -6.6321846664333062 1D-84,    9.1807650421280539 9D-88,
     E    -1.1856857861459452 4D-91,    1.4319876644274701 0D-95,
     F    -1.6208122970316582 9D-99,    1.7228021864707252 2D-103,
     G    -1.7229744839191171 3D-107,   1.6242217985662868 9D-111,
     H    -1.4456802835480969 2D-115,   1.2169025955792061 6D-119/

      DATA (D(I),I=1,36) /
     *          -2.2609957504809195D-001,    9.4208156270038315 5D-03,
     1    -1.4953675598418780 2D-04,    1.2461396332015650 2D-06,
     2    -6.3904596574439231 8D-09,    2.2189096032791399 9D-11,
     3    -5.5611709505742856 9D-14,    1.0532532103360389 6D-16,
     4    -1.5603751264237614 2D-19,    1.8575894362187635 9D-22,
     5    -1.8158254508492312 7D-25,    1.4835175252036203 2D-28,
     6    -1.0280786730447819 3D-31,    6.1195159109808448 1D-35,
     7    -3.1625405224707208 3D-38,    1.4323100192349279 1D-41,
     8    -5.7315326900157179 4D-45,    2.0411441203759679 3D-48,
     9    -6.5108265402742198 6D-52,    1.8709271667454654 8D-55,
     A    -4.8684027237717030 4D-59,    1.1525574630141342 4D-62,
     B    -2.4930942310493926 9D-66,    4.9466155377964140 7D-70,
     C    -9.0349142242856878 0D-74,    1.5241083374301092 8D-77,
     D    -2.3817914321458185 3D-81,    3.4578853544509560 6D-85,
     E    -4.6759774908058905 4D-89,    5.9040119833407708 9D-93,
     F    -6.9762637165789565 0D-97,    7.7307886930174606 6D-101,
     G    -8.0503891419529945 5D-105,   7.8925383744637201 4D-109,
     H    -7.2977701104611374 4D-113,   6.3747118365313919 0D-117/

      DATA(C4(I), I=1,19) /
     *        .10416666666666666666D000,     -.58763744212962962 94D000,
     *       -.22907160539343377 12D001,     -.51152469146043830 39D001,
     *       -.90628476638740308 39D001,     -.14134204350396378 96D002,
     *       -.20329678176117332 57D002,     -.27649485411187761 09D002,
     *       -.36093767125929491 87D002,     -.45662621146185479 16D002,
     *       -.56356118497383940 99D002,     -.68174312623273330 36D002,
     *       -.81117244833084638 49D002,     -.95184947011821749 27D002,
     *       -.11037744698909572 46D003,     -.12669488225846890 90D003,
     *       -.14413913537100938 69D003,     -.16273270737519703 76D003,
     *       -.18264442611464413 83D003/
      DATA(D4(I), I=1,19) /
     *       -.14583333333333333333D000,     -.52426938657407407 34D000,
     *       -.21900107400106261 22D001,     -.49862280807822504 17D001,
     *       -.89102698762513757 31D001,     -.13961075131923849 56D002,
     *       -.20138105970329288 47D002,     -.27441048801201192 11D002,
```

```
      *      -.35869703054434667370002,       -.45423931711946715330002,
      *      -.56103636448057570000002,       -.67908743957298515690002,
      *      -.80839198029929514380002,       -.94894955810820384810002,
      *      -.11007598935585749100003,       -.12638233054257933230003,
      *      -.14381582552263232700003,       -.16241253560510208670003,
      *      -.18264308105005668610003/
       DATA(C5(I), I=1,19) /
      *      -.80208333333333333220000,       -.22855450236966824530001,
      *      -.37786353593898852400001,       -.52746231607110598620001,
      *      -.67719261704049238570001,       -.82699549413402740750001,
      *      -.97684336200979706920001,       -.11267213747680068060002,
      *      -.12766207438002797210002,       -.14265358922464757740002,
      *      -.15764630887783273850002,       -.17263997303951796500002,
      *      -.18763439420525811750002,       -.20262943441407857240002,
      *      -.21762496686030703260002,       -.23262048426053788200002,
      *      -.24761042850013399850002,       -.26254300552690594930002,
      *      -.27720979241646002400002/
       DATA(D5(I), I=1,19) /
      *      -.67708333333333333220000,       -.22029147982062780150001,
      *      -.37125903088615039470001,       -.52180102894988778220001,
      *      -.67215926158079361730001,       -.82241865333529918370001,
      *      -.97261769556783931290001,       -.11227766967105787810002,
      *      -.12729075204990961410002,       -.14230176248921136010002,
      *      -.15731119656707000070002,       -.17231939820272207020002,
      *      -.18732661405573092580002,       -.20233302328193519550002,
      *      -.21733874940489829100002,       -.23234359923945613750002,
      *      -.24734046352953744400002,       -.26222528770880355710002,
      *      -.27586862583465528640002/
       DATA (ZMLA5(I), I=1,17) / 1.E9,715., 207., 103., 47., 36.4, 27.,
      * 22.6, 18.5, 16.6, 14.7, 14., 12.9, 12.2, 11.5, 10.8, 9.2 /
       DATA LA2, LA5 /36,17/
       DATA (ZMLA2(I),I=1,40) /
      12.6944301D-12,4.7348244D-6,7.0803713D-4,9.6398932D-3,
      24.9271494D-2,1.5267301D-1,3.5324772D-1,6.7835277D-1,
      31.1475215D0,1.7731141D0,2.5617749D0,3.9957181D0,5.2159327D0,
      46.6020702D0,8.1490972D0,9.8514112D0,1.2320405D1,1.4917923D1,
      51.7163634D1,1.9540309D1,2.3030246D1,2.6446844D1,2.9292820D1,
      63.3549744D1,3.7541246D1,4.0802980D1,4.5784933D1,5.0287133D1,
      75.3917166D1,5.9582285D1,6.4537858D1,7.0701377D1,7.5978472D1,
      88.0163399D1,8.6945766D1,9.2594255D1,9.983446D1,
      91.0575193D2,1.1039828D2,1.1820169D2/
       DATA   C1 / 0.57735 026918962576D0  /
       DATA   C2 / 0.66666 66666666666D0  /
       DATA   C3 / 0.86602 540378443864D0  /
       DATA   PI / 3.14159265358979324D0  /
       DATA FPI12 / 1.30899693899574718D0/
       DATA   CON4 / .70710678118654752440D0/
       A0 = 9.30436716929229427D-01
       B0 = 6.78298725144275871D-01
       C0 = 4.65218358464614714D-01
       D0 = 6.78298725144275871D-01
       K =    0.85366721883895156D0
       ZMSQ = ZR*ZR +ZI*ZI
       RZR = ZR
       TEMP = ZI
       IF (TEMP .LT. 0.) TEMP = -TEMP
       IF (RZR .LE. 0.) GO TO 51
```

```
      IF (RZR .GT. 4.4) GO TO 120
      TEM1 = 7. - .2632 * RZR**2
      IF (TEMP .GT. TEM1) GO TO 120
      GO TO 53
51    IF (RZR .LT. -9.) GO TO 120
      TEM1 = 4.4 + .1375 * RZR
      IF (TEMP .GT. TEM1) GO TO 120
53    FLPS = 1
      STORE1 = ZR*ZR-ZI*ZI
      STORE2 = 2.*ZR*ZI
      XR = STORE1*ZR -STORE2*ZI
      XI = STORE1*ZI +STORE2*ZR
      DO 55 MLS=1,LA2
      IF (ZMSQ - ZMLA2(MLS)) 62,62,55
55    CONTINUE
62    FR = A0
      FI = 0.0
      XPR(1) = XR
      XPI(1) = XI
      DO 65 M = 1,MLS
      FR=FR+A(M)*XPR(M)
      FI=FI+A(M)*XPI(M)
      XPR(M+1)=XR*XPR(M)-XI*XPI(M)
      XPI(M+1)=XI*XPR(M)+XR*XPI(M)
65    CONTINUE
      GR=B0
      GI=0.0
      DO 72 M = 1,MLS
      GR=GR+B(M)*XPR(M)
      GI=GI+B(M)*XPI(M)
72    CONTINUE
      X =ZR*GR-ZI*GI
      GI=ZR*GI+ZI*GR
      GR=X
      SR=-C1*(GI-FI-FI)
      SI=C1*(GR-FR-FR)
      H2R=GR-SR
      H2I=GI-SI
      GO TO 317
120   FLPS = 0
      ZM = DSQRT(ZMSQ)
      ZRT2M = DSQRT(ZM)
      IF (ZR .LT. 0.D0) GO TO 125
      ZRT2R = DSQRT (0.5D0 * (ZR + ZM))
      ZRT2I = ZI / (ZRT2R + ZRT2R)
      Z32R = ZR*ZRT2R - ZI*ZRT2I
      Z32I = ZR*ZRT2I + ZI*ZRT2R
      GO TO 130
125   ZRT2I = DSQRT (0.5D0 * (ZM - ZR))
      IF (ZI .LT. 0.D0) ZRT2I = -ZRT2I
      ZRT2R = ZI / (ZRT2I + ZRT2I)
      Z32R = ZR*ZRT2R - ZI*ZRT2I
      Z32I = ZR*ZRT2I + ZI*ZRT2R
      ZM1R = DABS(Z32I)
      IF (ZM1R .LT. TLIM) GO TO 130
      R = (TLIM / ZM1R)
      Z32R = Z32R * R
```

```
      R = DCBRT(R)
      ZRT2R = ZRT2R * R
      ZRT2I = ZRT2I * R
      ZRT2M = ZRT2M * R
      R = R * R
      ZM = ZM * R
      ZMSQ = ZM**2
      ZR = ZR * R
      ZI = ZI * R
130   ZRT4R = DSQRT (0.5D0 * (ZRT2R + ZRT2M))
      ZRT4I = 0.5D0 * ZRT2I / ZRT4R
      ZRTM4R = ZRT4R/ZRT2M
      ZRTM4I = -ZRT4I/ZRT2M
      IF (ZR .GT. 0.) GO TO 210
      IF (ZM1R .LT. TLIM) GO TO 210
      ABK = ABS(K2)
      IF (Z32I .GT. 0.) GO TO 205
      K1 = K * EXPONT
      K2 = K / EXPONT
      Z32I = -TLIM
      GO TO 220
205   K2 = K * EXPONT
      K1 = K / EXPONT
      Z32I = TLIM
      GO TO 220
210   K2 = C2 * Z32I
      S2 = DEXP(K2)
      K2 = K*S2
      K1 = K/S2
220   THR = FPI12 - C2 * Z32R
      STHR =DSIN(THR)
      CTHR =DCOS(THR)
      STP = -C3*CTHR +0.5*STHR
      CTP = -C3*STHR -0.5*CTHR
      TEMP = DABS (Z32R)
      TEM1 = DABS (Z32I)
      IF (TEMP .LT. TEM1) TEMP = TEM1
230   DO 235  ML = 1,LA5
      IF (TEMP .GT. ZMLA5(ML)) GO TO 250
235   CONTINUE
250   CONTINUE
      YR = Z32I
      YI = -Z32R
      CALL CFR (YR, YI, F2R, F2I, C4, C5, ML)
      CPR = F2R
      CPI = F2I
      STORE3=K2*(ZRTM4R*F2R-ZRTM4I*F2I)
      STORE4=K2*(ZRTM4I*F2R+ZRTM4R*F2I)
      H22R =STORE3*CTHR-STORE4*STHR
      H22I =STORE3*STHR+STORE4*CTHR
      IF (ZR) 280,270,270
270   FLQUAD =0
      GO TO 300
280   IF (ZI) 290,310,310
290   FLQUAD = 1
300   H2R = H22R
      H2I = H22I
```

```
      GO TO 317
310   FLQUAD = -1
      YR = -Z32I
      YI = Z32R
      CALL CFR (YR, YI, F1R, F1I, C4, C5, ML)
      CPR = F1R
      CPI = F1I
      STORE5=K1*(ZRTM4R*F1R-ZRTM4I*F1I)
      STORE6=K1*(ZRTM4R*F1I+ZRTM4I*F1R)
      H21R=STORE5*CTP-STORE6*STP
      H21I=STORE5*STP+STORE6*CTP
      H2R=H21R+H22R
      H2I=H21I+H22I
317   IF (IH .EQ. 2)GO TO 80
60    IF (FLPS .NE. 1) GO TO 320
70    H1R = GR+SR
      H1I = GI+SI
      GO TO 362
320   IF (FLQUAD .LT. 0)GO TO 340
330   YR = -Z32I
      YI = Z32R
      CALL CFR (YR, YI, F1R, F1I, C4, C5, ML)
340   STORE7=K1*(ZRTM4R*F1R-ZRTM4I*F1I)
      STORE8=K1*(ZRTM4I*F1R+ZRTM4R*F1I)
      H11R=STORE7*CTHR+STORE8*STHR
      H11I=STORE7*(-STHR)+STORE8*CTHR
      IF (FLQUAD .LE. 0) GO TO 360
350   STORE9=K2*(ZRTM4R*F2R-ZRTM4I*F2I)
      STOR10=K2*(ZRTM4I*F2R+ZRTM4R*F2I)
      H12R = STORE9*CTP+STOR10*STP
      H12I = STORE9*(-STP)+STOR10*CTP
      H1R = H11R+H12R
      H1I = H11I+H12I
      GO TO 362
360   H1R = H11R
      H1I = H11I
362   IF (IH .EQ. 1)GO TO 999
80    IF (FLPS .NE. 1) GO TO 380
90    FPR = C0
      FPI = 0.0
      DO 92 M = 1,MLS
      FPR=FPR+C(M)*XPR(M)
92    FPI=FPI+C(M)*XPI(M)
      X  =-(STORE1*FPR-STORE2*FPI)
      FPI=-(STORE1*FPI+STORE2*FPR)
      FPR = X
      GPR = D0
      GPI = 0.0
      DO 94 M = 1,MLS
      GPR=GPR+D(M)*XPR(M)
94    GPI=GPI+D(M)*XPI(M)
      SPR=-C1*(GPI-FPI-FPI)
      SPI=C1*(GPR-FPR-FPR)
      H2PR=GPR-SPR
      H2PI=GPI-SPI
      GO TO 414
380   YR =  Z32I
```

```
           YI = -Z32R
           CALL CFR (YR, YI, G2R, G2I, D4, D5, ML)
           STOR11 = K2*(ZRT4R*G2R-ZRT4I*G2I)
           STOR12 = K2*(ZRT4R*G2I+ZRT4I*G2R)
           H22PR=STOR11*STHR+STOR12*CTHR
           H22PI=STOR11*(-CTHR) +STOR12*STHR
     390   IF (FLQUAD .LT. 0) GO TO 410
     400   H2PR = H22PR
           H2PI = H22PI
           GO TO 414
     410   YR = -Z32I
           YI = Z32R
           CALL CFR (YR, YI, G1R, G1I, D4, D5, ML)
           STOR13 = K1*(ZRT4R*G1R-ZRT4I*G1I)
           STOR14 = K1*(ZRT4R*G1I+ZRT4I*G1R)
           H21PR=STOR13*(-STP) -STOR14*CTP
           H21PI=STOR14*(-STP) +STOR13*CTP
           H2PR = H21PR+H22PR
           H2PI = H21PI+H22PI
     414   IF (IH .EQ. 2) GO TO 999
     100   IF (FLPS .NE. 1) GO TO 420
     110   H1PR = GPR+SPR
           H1PI = GPI+SPI
           GO TO 999
     420   IF (FLQUAD .LT. 0) GO TO 440
     430   YR = -Z32I
           YI = Z32R
           CALL CFR (YR, YI, G1R, G1I, D4, D5, ML)
     440   STOR15 = K1*(ZRT4R*G1R -ZRT4I*G1I)
           STOR16 = K1*(ZRT4R*G1I +ZRT4I*G1R)
           H11PR = STOR15*STHR -STOR16*CTHR
           H11PI = STOR15*CTHR +STOR16*STHR
     450   IF (FLQUAD .GT. 0) GO TO 470
     460   H1PR = H11PR
           H1PI = H11PI
           GO TO 999
     470   STOR17 = K2*(ZRT4R*G2R -ZRT4I*G2I)
           STOR18 = K2*(ZRT4R*G2I +ZRT4I*G2R)
           H12PR = STOR17*(-STP) +STOR18*CTP
           H12PI = STOR17*(-CTP) -STOR18*STP
           H1PR = H12PR+H11PR
           H1PI = H12PI+H11PI
     999   CONTINUE
           RETURN
           END

@PRT,S  J.CFR
```

```
      SUBROUTINE CFR(X, Y, SR, SI, A, B, M)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(1), B(1)
      SR = 0.D0
      SI = 0.D0
      DO 10 J = 1,M
      I = M - J + 1
      TEMR = X + SR + B(I)
      TEMI = Y + SI
      TEMP = A(I) / (TEMR**2 + TEMI**2)
      SR = TEMR * TEMP
      SI = -TEMI * TEMP
   10 CONTINUE
      SR = SR + 1.D0
      RETURN
      END
```

```
      SUBROUTINE HZERO(PARTR,PARTI)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION PARTI, PARTR, K2, IMZ12
      COMMON /AHZERO/ HZEROR,HZEROI
      D2 = PARTR**2 + PARTI**2
    8 K2 = .7978845608*EXP(PARTI)
      D = SQRT(D2)
      RLZ12 = (SQRT((PARTR + D)/2.))/D
      IF(D - PARTR) 9,9,10
    9 IMZ12 = 0
      GO TO 11
   10 IMZ12 = (SQRT((D - PARTR)/2.))/D
   11 COST = COS(PARTR)
      SINT =-SIN(PARTR)
      HZEROR = K2*(RLZ12*COST - IMZ12*SINT)
      HZEROI = K2*(IMZ12*COST + RLZ12*SINT)
      RETURN
      END
```

APPENDIX B: SAMPLE RUN

This appendix gives a brief discussion of the input-output, then lists an input deck and shows the resulting output. The input deck is really three separate runs that are stacked to run consecutively. The input to a single run consists of several parts given in table B1. The table gives the number of cards and the location of the FORTRAN input statements in Program MAIN. The last three of these are open-ended. That is, more modes, receiver depths, or ranges are read in until a blank card specifies the end of the list. A blank range card sends the program back to the beginning. A negative range sends the program back to read a new source and new receivers after reading another key card. The program halts when a blank "n and freq" card is encountered.

Table B2 gives most of the functions of the key card by which integers are read into control keys 1-9. Some of these will require additional information, which is read in immediately following the key card.

The output of the program is usually printed through FORTRAN print statements. Cards are also punched when key 5 is 10 or key 8 is 2, 3, or 4. In the first case each card contains a complete eigenvalue that can be read into future runs.

When key 8 = 2, propagation losses for 25 consecutive ranges per card are punched for each receiver depth, with a maximum of 5 receiver depths. The losses can be read into a plot program with a format of (5X,25F3.1). Each loss must then be subtracted from 140. This format allows losses to tenths of a dB from 40.1 to 140.0 dB.

When key 8 is 3, losses for up to 26 receiver depths are punched on one card for each range. These cards can be used in a contour plotting program. They can be read with a format of (26F3.1) and must also be subtracted from 140.

Table B1. Input cards to the normal mode program.

Input	Function	Number of Cards	Location in Program MAIN
Control keys	Selects options	1 or more	37-65
n and freq	Determines number of layers and frequency; also halts program	1	66
Profile	Specifies depths, sound speeds, gradients, attenuations, and densities	7	71-85
Modes	Searches for or specifies modes	1 or more	224
Source and receivers	Specifies a source depth and one or more receiver depths	2 or more	461-463
Ranges	Specifies a sequence of ranges; also directs continuation	1 or more	616

Table B2. Functions of the control keys.

Key	Setting	Effect	Function Affected
1	0	No output	Depth functions
		Print	
		Plot on printer	
2	0	Print losses	Propagation losses
	1	Contour on printer	
3	0	No output	Reflection coefficients
	1	Print all interfaces	
	k > 1	Print interface k	
4	k > 0	Change levels and symbols	Contour on printer
5	0	Sum only those given	Number of modes
	1	Add to existing sum	
	k + 10	Punch modes on cards	
6	0	Long print	Steps in mode iteration
	1	Short print	
	2	Shortest print	
7	0	Phased addition	Mode sum
	1	Random-phase addition	
8	1	Change T-lim	
	2	Punch losses for up to 5 receivers	Loss plot input
	3	Punch losses for up to 26 receivers	Contour plot input
9	0	No effect	Number of modes
	k	Use only 1st k modes	

The first profile in the input-output is a surface duct, 100 m deep. For the 500 Hz frequency, 3 modes are found by searching from a phase velocity of 1520.5–1523 m/s. Two additional modes are found by extrapolation. Forty receiver depths are then specified from 3 to 120 m, and propagation loss contours are drawn for a source at a depth of 20 m. The modes are added in random phase, and loss contours of 80, 90, and 100 dB are requested to be represented by the symbols 8, 9, and 0. A negative range then causes the program to read new control keys, source and receivers. The depth functions are then printed out as amplitudes and phase angles and propagation losses are computed.

The second profile consists of two negative gradients over two layers of sediments in shallow water. A velocity discontinuity exists at the top of each sediment layer. Negative numbers in the input for the attenuation at the bottom of the sediment layers serve as flags to request that the gradients at the top of the layers have no imaginary parts and that the attenuation at the bottom will be whatever results from this. The change in ImC from 37.9 to 23.7 in the deeper sediment layer indicates that the attenuation changed by about 60 percent through the layer.

A final layer of negative sound speed gradient must always be added. A gradient of –0.1 is chosen here for the top of this layer.

The first three modes are determined by reading in approximate values. The magnitudes of the depth functions are plotted on a log scale at 2-m depths from 30 to 80 m. Reflection coefficients are computed at interface 2, which is the water-sediment interface.

The final profile is a deep-water profile with a 40-m deep sediment layer. The attenuation increases from 2 dB/km to 2.5 dB/km through this layer. The first mode, the first bottom-reflected mode, and a higher bottom-reflected mode are found. Each step of the mode iteration is printed out. Reflection coefficients are again computed. The amplitudes and phase of the depth functions are printed out at 500-m depth and at each even 1000-m depth for a 100-m source depth.

On the last two profiles, a much larger set of modes is required to compute correct propagation losses.

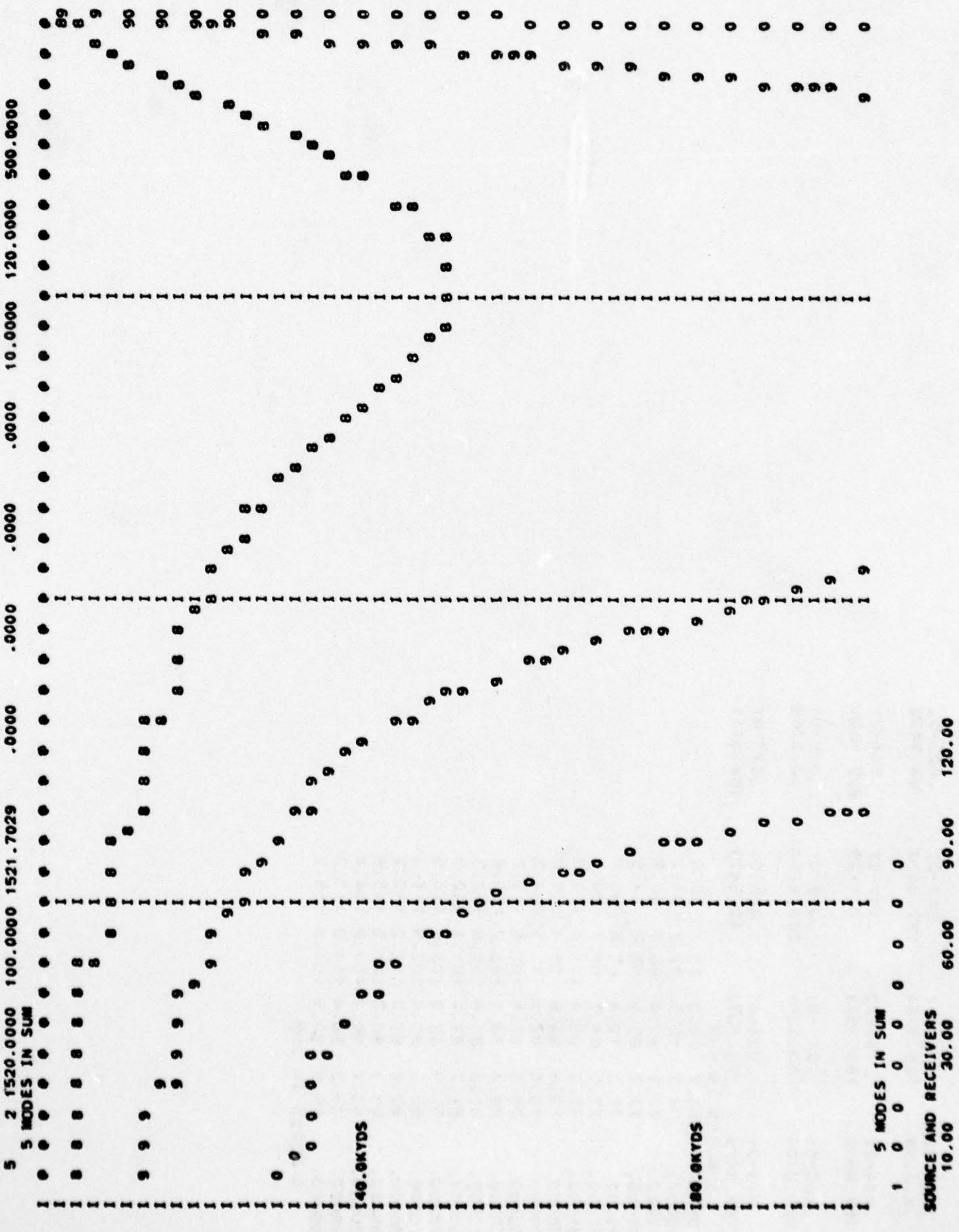
The sample run given here required 6 seconds on a UNIVAC 1110, Exec 8 operating system. The cost of the run was \$1.20.

```
NORMAL*MODE(0).INPUT
     1     INPUT DECK STARTS AT LINE 3,  ENDS AT LINE 65.
     2    123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
     3           1       1       1   1   2
     4    100.      90.       80.       -1000.
     5    0
     6    098
     7     2  500.
     8              100.
     9    1520.
    10    0
    11    .017      -.1
    12    0
    13    0
    14    0
    15    -1520.5             1523.                                   30
    16    -1.                                                          2
    17    0
    18    10.
    19    3.        120.      3.
    20    0
    21    4000.     100000.   4000.
    22    -1.
    23       1
    24    10.
    25    30.       120.      30.
    26    0
    27    5000.     100000.   5000.
    28    0
    29       2       2           1
    30     5  1500.
    31    0.        51.       73.       73.3      373.3
    32    1542.2    1536.8    1606.45   1684.
    33              1523.42
    34                        1.5       1.5       -.1
    35                        .12       .73       .73
    36                        -1.       -1.
    37                        1.68      1.91      1.91
    38    1527.18   .16
    39    1530.64   .13
    40    1533.49   .11
    41    0
    42    60.
    43    30.       80.       2.
    44    0
    45    0
    46     1        6
    47     8  100.
    48              55.       146.      402.      960.      2286.     4390.     4430.
    49    1544.9    1542.6    1517.9    1495.0    1483.2    1497.8    1541.7
    50                                                      1533.4
    51                                                                1.        -.1
    52                                                                .02       .025
    53                                                                .025
    54                                                                1.54      2.5
    55    -1483.5             1484.5                          10
    56    -1533.4   .1        1534.4                          10
```

57	-1600.	.2	1602.		10			
58	0							
59	100.							
60	500.							
61	1000.	5000.	1000.					
62	0							
63	0							
64	0							
65	0							
66	123456789	123456789	123456789	123456789	123456789	123456789	123456789	123456789

```
KEYS     0   1   0   1   0   1   1   2   0
 2 LAYERS,      500.0 HZ
        .00000      100.00000
   1520.00000          .00000
        .00000         .00000
        .01700       -.10000
        .00000         .00000
        .00000         .00000
        .00000         .00000
ATTEN =.24539-001DB/KM
     RE M           IM M           L/KM          RE C            IM C        RE C BOTTOM     IM C BOTTOM           GRADIENT
     4.57175        .00000        .00000     1520.00000        .00000     1521.70286        .00000        .01700        .00000
     8.24353        .00000        .00000     1521.70286        .00000         .00000        .00000       -.10000        .00000
   1520.500        .000      .16061+003      .10879+002     .65010+001
   1520.583        .000      .48580+002      .59690+001     .35987+001
   1520.667        .000      .12071+002      .29662+001     .18090+001
   1520.750        .000      .20274+001      .12104+001     .74991+000
   1520.833        .000      .88666-001      .25086+000     .16043+000
   1520.917        .000      .65065-001     -.21599+000    -.13569+000
   1521.000        .000      .21841+000     -.39129+000    -.25554+000

   .15209166667+004     .00000000000          -.2160+000    -.1357+000     .2184+000     .0000
   .15208699858+004     .36285586937-003      -.9521-014    -.6003-014     .5345-022     .0000
Z =    233     0    -68     0
Z =   -223     0     0      0
 .000  .000   .000  .000
N=     1  LAMBDA =     .2065655+01    -.4928331-06 D=    -.4937188+02     .1698935+00 INT RANGE =      0. L/K =  .00428
   1520.953        .000      .14432+000     -.31942+000    -.20563+000
   1521.037        .000      .24401+000     -.41150+000    -.27328+000
   1521.120        .000      .21725+000     -.38317+000    -.26539+000
   1521.203        .000      .13931+000     -.30074+000    -.22105+000
   1521.287        .000      .68182-001     -.20341+000    -.16373+000
   1521.370        .000      .24079-001     -.11266+000    -.10671+000
   1521.453        .000      .47194-002     -.38741-001    -.56732-001
   1521.537        .000      .49825-003      .14984-001    -.16545-001
   1521.620        .000      .25922-002      .49104-001     .13453-001

   .15215366524+004     .36285586937-003       .1498-001    -.1655-001     .2592-002     .0000
   .15215320352+004     .33379340870-001       .3642-015     .2081-015     .1058-020     .0000
Z =    411    89    -14    27
Z =    -45    89     0      0
 .000  .000   .000  .000
N=     2  LAMBDA =     .2064756+01    -.4529658-04 D=    -.4419460+02     .7013564+01 INT RANGE =   6991. L/K =  .39345
   1521.615        .033      .20449-002      .38891-001     .23074-001
   1521.699        .033      .50536-002      .60096-001     .37975-001
   1521.782        .033      .66370-002      .67011-001     .46331-001
   1521.865        .033      .64766-002      .63289-001     .49711-001
   1521.949        .033      .51912-002      .52365-001     .49488-001
   1522.032        .033      .35740-002      .37214-001     .46789-001
   1522.115        .033      .22152-002      .20244-001     .42489-001
   1522.199        .033      .13974-002      .32976-002     .37236-001
   1522.282        .033      .11427-002     -.12314-001     .31482-001
   1522.365        .033      .13140-002     -.25737-001     .25527-001

   .15222820352+004     .33379340870-001      -.1231-001     .3148-001     .1314-002     .0000
   .15222575167+004     .21101619027+000       .6334-016     .2739-017     .1810-020     .0000
Z =    6[illegible]   564    45   173
```

```
Z =   148  564    0    0
 .000 .000  .000 .000
N=    3  LAMBDA =     .2063772+01    -.2860812-03 D=    -.2457285+02    .8311493+01 INT RANGE =   6385. L/K = 2.48490
    1522.341          .211       .23202-003       -.15210-001       -.82123-003
    1522.424          .211       .75319-003       -.27433-001       -.77326-003
    1522.508          .211       .13283-002       -.36444-001       -.39025-003
    1522.591          .211       .17893-002       -.42300-001       -.92158-004

   .15230464302+004    .53327340407+000   -.2374-001    .1860-001    .1789-002    .0000
   .15233048329+004    .50934840311+000    .8377-015   -.3968-015    .3366-024    .0000
Z =   885 1360    131  418
Z =   428 1360      0    0
 .000 .000  .000 .000
N=    4  LAMBDA =     .2062353+01    -.6895903-03 D=    -.1425623+02    .5114411+01 INT RANGE =   4428. L/K = 5.98978

   .15246739840+004    .92837597939+000   -.7307-002   -.2996-003    .3366-024    .0000
   .15247283934+004    .88008242577+000    .1040-014   -.1501-014    .1148-016    .0000
Z =  1265 2344    248  721
Z =   808 2344      0    0
 .000 .000  .000 .000
N=    5  LAMBDA =     .2060427+01    -.1189291-02 D=    -.9215171+01    .3160372+01 INT RANGE =   3263. L/K =10.33018

SOURCE AND RECEIVERS
    10.00      3.00      6.00      9.00     12.00     15.00     18.00     21.00
    24.00     27.00     30.00     33.00     36.00     39.00     42.00     45.00
    48.00     51.00     54.00     57.00     60.00     63.00     66.00     69.00
    72.00     75.00     78.00     81.00     84.00     87.00     90.00     93.00
    96.00     99.00    102.00    105.00    108.00    111.00    114.00    117.00
   120.00
```


5 2 1520.0000 100.0000 1521.7029 .0000 .0000 .0000 10.0000 120.0000 500.0000
5 MODES IN SUM
40.0KYDS
80.0KYDS
1 5 MODES IN SUM
SOURCE AND RECEIVERS
10.00 30.00 60.00 90.00 120.00

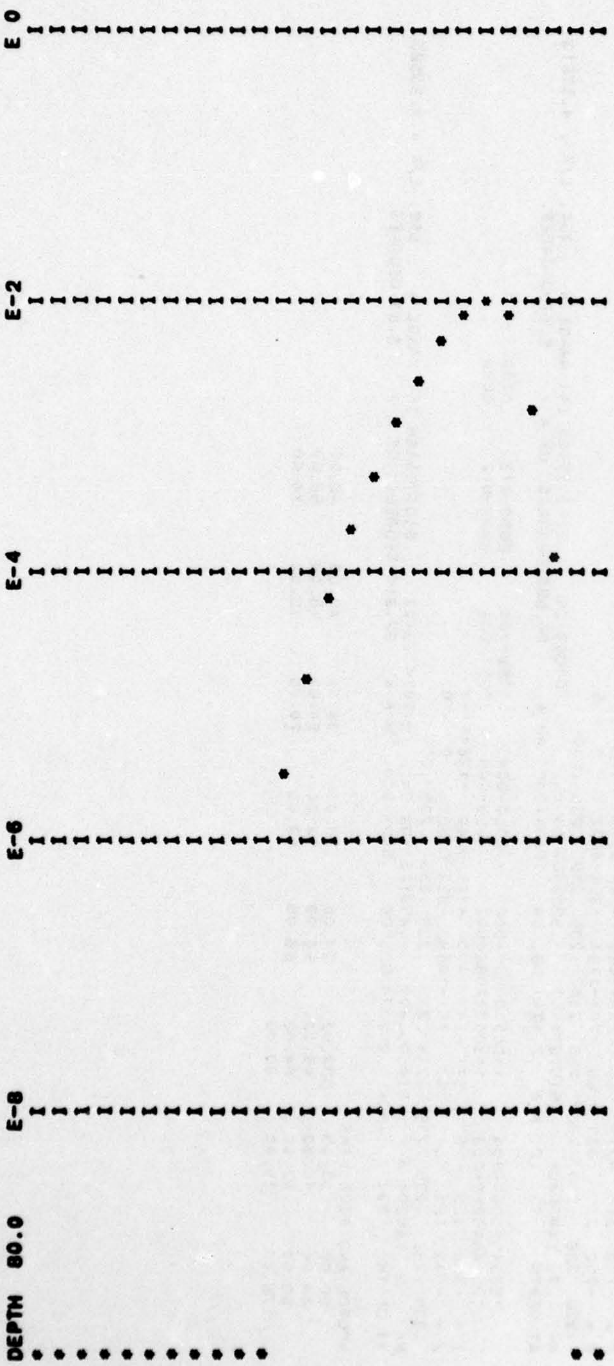
.151-01	.716-02	.143-02	.448-03
.1725	-.0318	-4.0388	-78.8365
.625-02	.128-01	.847-02	.403-02
5.3138	188.4864	172.8414	66.9433
.244-02	.768-02	.197-01	.146-01
269.9452	150.1511	2.1129	222.9555
.138-01	.221-01	.316-01	.358-01
209.0704	22.5897	201.6754	14.5748
.213-01	.221-01	.455-01	.677-01
186.2405	-70.1573	42.6473	159.3411

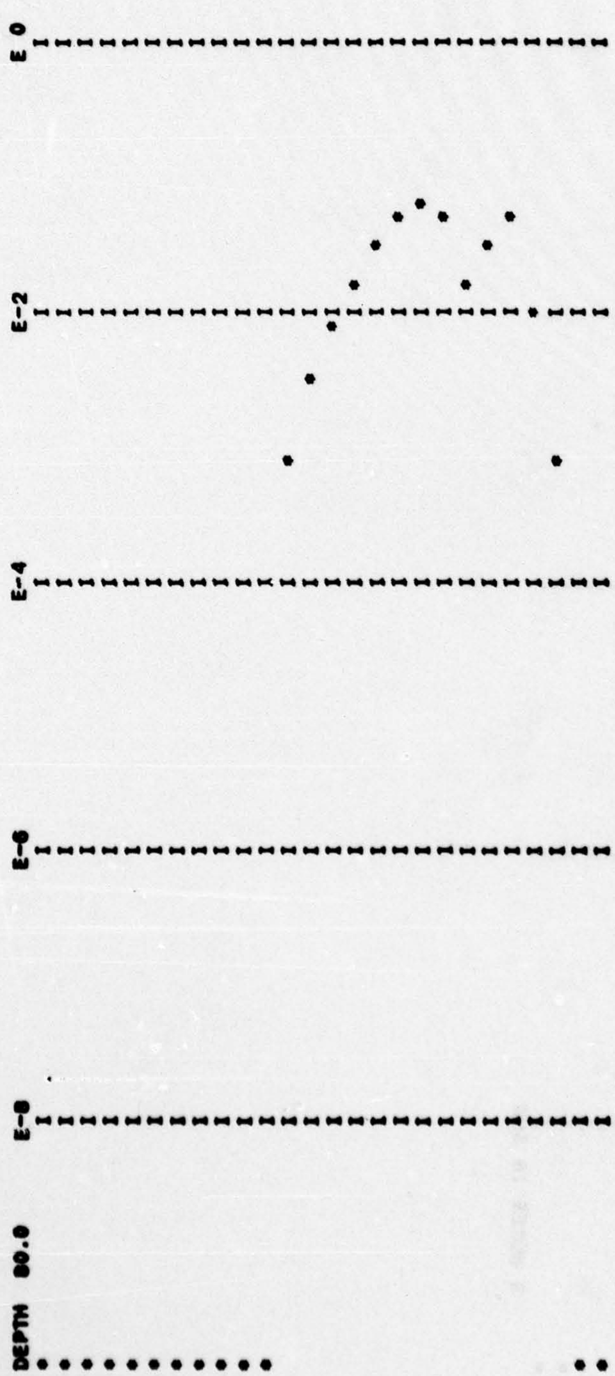
5 MODES IN SUM

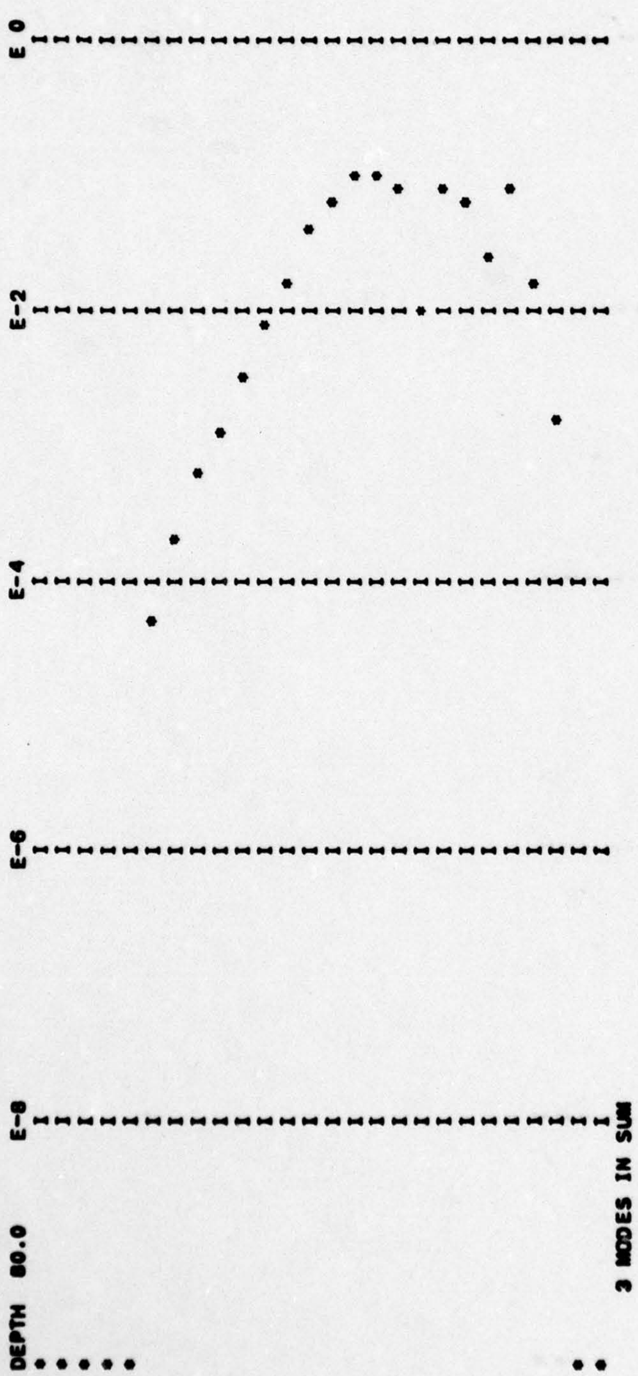
5000.	68.7	70.3	75.1	78.8
10000.	74.3	72.1	77.7	84.2
15000.	72.7	80.4	86.0	92.9
20000.	74.2	85.2	89.7	94.5
25000.	77.3	79.2	87.6	95.0
30000.	77.5	81.7	91.7	100.8
35000.	77.3	88.4	111.0	107.3
40000.	78.8	84.8	95.4	102.0
45000.	79.9	84.1	95.1	104.2
50000.	79.8	87.1	102.6	119.6
55000.	80.4	88.1	103.2	109.8
60000.	81.3	86.8	98.8	107.5
65000.	81.7	87.7	101.1	112.3
70000.	81.9	89.2	105.1	115.9
75000.	82.5	89.0	102.5	111.3
80000.	83.0	89.0	102.2	112.2
85000.	83.3	90.0	104.5	115.6
90000.	83.7	90.4	104.8	114.4
95000.	84.2	90.4	104.0	113.6
100000.	84.5	90.9	104.8	115.2

5 MODES IN SUM

```
KEYS    2    0    2    0    0    1    0    0    0
 5 LAYERS.      1500.0 HZ
        .00000      51.00000      73.00000      73.30000     373.30000
    1542.20000    1536.80000    1606.00000    1684.00000       .C0000
        .00000    1523.42000       .00000       .00000       .C0000
        .00000       .00000      1.50000      1.50000      -.10000
        .00000       .00000       .12000       .73000       .73000
        .00000       .00000     -1.00000     -1.00000       .C0000
        .00000       .00000      1.68000      1.91000      1.91000
ATTEN =.99703-001DB/KM
       RE M           IM M          L/KM          RE C          IM C      RE C BOTTOM    IM C BOTTOM           GRADIENT
    -8.81029       .00000       .00000    1542.20000       .C0000    1536.80000       .00000     -.10644       .00000
    -6.84816       .00000       .00000    1536.80000       .C0000    1523.42000       .00000     -.61622       .00000
      .12021      -.00042    180.00229    1606.00000      5.67131    1606.45019      5.67131     1.50000       .00000
   114.58114     -2.58229   1095.01391    1684.00000     37.95189    2467.43825     23.72336     1.50000       .00000
    39.43319     -1.30290   1095.01391    2467.43825     81.52581       .00000       .00000     -.10000       .00000

   .15271800000+004    .16000000000+000    -.2525-004     .2950-004     .1148-016     .0000
   .15271861943+004    .16198647471+000     .1348-008    -.6792-008     .1197-013     .0000
Z =  -750   267  -490    83  -749-1469  -749-9189  -177*****
Z =  -750   267   194    83  -761-1469  -854-6607     0     0
 .237 .237  .237 .237  .237 .237  .237 .237  .237 .000
N=    1  LAMBDA =     .6171335+01    -.6545848-03 D=    -.2850630+C7    .1058118+07 INT RANGE =      0. L/K = 5.68572
AT DEPTH     73. YD, R =   1.9028 DB, OR  .6452+000, PH A =  113.538 DEGREES, GR A =    4.02 DEGREES.

   .15306400000+004    .13000000000+000     .1114-005    -.2117-005     .1197-013     .0000
   .15306495580+004    .13096622218+000     .6485-010     .2047-005     .2050-015     .0000
Z =  -750   215  -312    66  -749-1482  -749-9219  -176*****
Z =  -750   215   372    66  -761-1481  -854-6637     0     0
 .236 .236  .236 .236  .236 .236  .236 .236  .236 .000
N=    2  LAMBDA =     .6157371+01    -.5268402-03 D=    -.4104669+C4    .9252135+03 INT RANGE =    450. L/K = 4.57613
AT DEPTH     73. YD, R =   2.1761 DB, OR  .6059+000, PH A =   96.596 DEGREES, GR A =    5.57 DEGREES.

   .15334900000+004    .11000000000+000    -.2482-004     .1760-004     .2050-015     .0000
   .15334940269+004    .11306008086+000    -.4059-008     .3391-008     .8069-012     .0000
Z =  -750   185  -167    57  -749-1490  -749-9242  -176*****
Z =  -543   185   517    57  -761-1489  -854-6660     0     0
 .235 .235  .235 .235  .235 .235  .235 .235  .235 .000
N=    3  LAMBDA =     .6145950+01    -.4531231-03 D=    -.2074773+C7    .6103755+06 INT RANGE =    550. L/K = 3.93583
AT DEPTH     73. YD, R =   2.2234 DB, OR  .5993+000, PH A =   87.310 DEGREES, GR A =    6.57 DEGREES.

SOURCE AND RECEIVERS
     60.00     30.00     32.00     34.00     36.00     38.00     40.00     42.00
     44.00     46.00     48.00     50.00     52.00     54.00     56.00     58.00
     60.00     62.00     64.00     66.00     68.00     70.00     72.00     74.00
     76.00     78.00     80.00
```


DEPTH 80.0
E-8
E-6
E-4
E-2
E 0


E 0
E-2
E-4
E-6
E-8
DEPTH 80.0


DEPTH 80.0
3 MODES IN SUM
E 0
E-2
E-4
E-6
E-8

```
KEYS    1   0   6   0   0   0   0   0   0
 8 LAYERS,      100.0 HZ
        .00000      55.00000     146.00000     402.00000     960.C0000    2286.00000    4390.00000    4430.00000
    1544.90000    1542.60000    1517.90000    1495.00000    1483.20000    1497.80000    1541.70000        .00000
        .00000        .00000        .00000        .00000        .C0000    1533.40000        .00000        .00000
        .00000        .00000        .00000        .00000        .C0000        .00000       1.00000      -.10000
        .00000        .00000        .00000        .00000        .C0000        .00000        .02000        .02500
        .00000        .00000        .00000        .00000        .C0000        .00000        .02500        .00000
        .00000        .00000        .00000        .00000        .C0000        .00000       1.54000       2.50000
ATTEN =.11895-002DB/KM
      RE M          IM M          L/KM          RE C          IM C     RE C BOTTOM   IM C BOTTOM               GRADIENT
      -1.14298        .00000        .00000    1544.90000        .C0000    1542.60000        .00000       -.04191        .00000
      -3.55888        .00000        .00000    1542.60000        .C0000    1517.90000        .00000       -.27809        .00000
      -7.02474        .00000        .00000    1517.90000        .C0000    1495.00000        .00000       -.09152        .00000
      -9.57769        .00000        .00000    1495.00000        .C0000    1483.20000        .00000       -.02140        .00000
      18.29300        .00000        .00000    1483.20000        .C0000    1497.80000        .00000        .01085        .00000
      32.94320        .00000        .00000    1497.80000        .C0000    1533.40000        .00000        .01634        .00000
       2.39805        .00378       2.00003    1541.70000        .E7104    1583.32718       1.14838       1.00000        .00642
     120.03173       -.08706       2.50003    1583.32718       1.14838        .00000        .00000       -.10000        .00000
     1483.500      .000       .31454+039       -.31203+019       -.17458+020
     1483.600      .000       .36434+039       -.33568+019       -.18790+020
     1483.700      .000       .38673+039       -.34568+019       -.19359+020
     1483.800      .000       .36481+039       -.33558+019       -.18803+020
     1483.900      .000       .28845+039       -.29827+019       -.16720+020
     1484.000      .000       .16645+039       -.22647+019       -.12701+020
     1484.100      .000       .42001+038       -.11371+019       -.63803+019
     1484.200      .000       .64709+037        .44612+018        .25044+019
     1484.300      .000       .20299+039        .24975+019        .14027+020

   .14842000000+004    .00000000000           .4461+018     .2504+019     .2030+039    .0000
   .14841653113+004    .15121756017-005      -.1565+018    -.8775+018     .2030+039    .0000
   .14841673537+004    .87643211278-006      -.1225+018    -.6873+018     .4171-005    .0000
   .14841747321+004    .23241154946-006       .1625+016     .9251+016     .5444-004    .0000
   .14841746342+004    .10000000000-017      -.1691+014    -.9489+014     .9596-008    .0000
   .14841746352+004    .24600381520-011      -.2477+010    -.1254+011     .9900-012    .0000
   .14841746352+004    .16732722238-014      -.1782+006    -.7745+005     .1743-019    .0000
Z =  -750     0  -750     0  -750     0  -750     0   123     0  -750.    0  -365   -40  -749  -307
Z =  -750     0  -514     0  -343     0    79     0  -854     0  -854     0  -604   -44     0     0
 .235  .235  .235  .235  .235  .235  .235  .235  .235  .235  .235  .235  .235  .235  .235  .000
N=    1  LAMBDA =     .4233454+00    -.4772835-18 D=    -.1233262+28    -.1275428+23 INT RANGE =      0. L/K =  .00000
AT DEPTH  4390. YD, R =  -7.7133 DB, OR   .5907+001, PH A =    -.098 DEGREES, GR A =     1.00 DEGREES.
     1484.275      .000       .12168+039        .19339+019        .10860+020
     1533.400      .100       .42744-001       -.94064-001        .18411+000
     1533.500      .100       .60899-001        .28875-001        .24508+000
     1533.600      .100       .46658-001        .12032+000        .17995+000
     1533.700      .100       .21506-001        .14180+000        .37415-001
     1533.800      .100       .19682-001        .93327-001       -.10475+000
     1533.900      .100       .32457-001        .73283-002       -.18001+000

   .15338000000+004    .10000000000+000       .9333-001    -.1047+000     .3246-001    .0000
   .15337624954+004    .10000000000-017      -.3138-002    -.1881-001     .3246-001    .0000
   .15337554931+004    .10000000000-017      -.2196-002    -.1055-001     .1589-003    .0000
   .15337464534+004    .83226880921-003       .4164-004    -.1464-003     .8241-004    .0000
   .15337465782+004    .81179352756-003      -.4671-006    -.5031-006     .1599-007    .0000
   .15337465774+004    .81209033462-003      -.6436-010    -.5979-010     .7043-012    .0000
   .15337465774+004    .81209038169-003      -.7988-014     .1180-013     .5258-020    .0000
Z =  -559     0  -125     0   467     0  2990     0  6112     0  3325.    0   -48   -50  -750  -317
```

```
Z =  -444    0  230    0 1170    0  3948    0  4283    0    30    0  -287  -54    0    0
  .235 .235  .235 .235  .235 .235  .235 .235  .235 .235  .235 .235  .235 .235  .235 .000
N=   2  LAMBDA =    .4096625+00    -.2169087-06 D=     -.7141127+C6    .7150644+03 INT RANGE =     459.  L/K =  .00188
AT DEPTH  4390.  YD,  R =    .0925 DB,  OR    .9789+000,  PH A =  153.731 DEGREES,  GR A =    1.22 DEGREES.
     1533.847         .001         .10792-001        -.11971-001         -.10319+000
     1533.947         .001         .21413-001        -.16789-001         -.14537+000
     1534.047         .001         .13258-001        -.13092-001         -.11440+000
     1534.147         .001         .11330-002        -.36517-002         -.33462-001
     1600.000         .200         .60957-020         .77953-010          .43583-011
     1600.200         .200         .62615-020         .77626-010         -.15354-010
     1600.400         .200         .29078-020         .47172-010         -.26128-010
     1600.600         .200         .37426-021         .43201-012         -.19341-010
     1600.800         .200         .18268-020        -.42571-010          .38020-011

     .16006000000+004      .20000000000+000      .4320-012    -.1934-01C    .1827-020    .0000
     .16006271212+004      .86853704778-001      .3449-011     .7868-011    .1827-020    .0000
     .16006062469+004      .92539871678-001      .7350-011     .5081-011    .4681-003    .0000
     .16006293949+004      .12555225080+000     -.1323-012    -.8078-013    .1626-002    .0000
     .16006289693+004      .12501210171+000      .3082-014    -.2074-015    .4729-006    .0000
     .16006289822+004      .12501608013+000      .8120-018    -.8123-018    .1818-009    .0000
     .16006289822+004      .12501607809+000      .4661-023     .7166-023    .2514-016    .0000
Z =  2620    55   772    15  2291    31  7652    81 13327   126  8926    98   333  -56   461 -271
Z =  2734    55  1128    15  2993    31  8609    81 11498   126  5632    98    94  -59     0    0
  .235 .235   .235 .235   .235 .235   .235 .235   .235 .235   .235 .235   .235 .235   .235 .000
N=   3  LAMBDA =      .3925448+00    -.3065945-04 D=     -.1425299+C3    .5003842+00 INT RANGE =     367.  L/K =  .26631
AT DEPTH  4390.  YD,  R =    6.3696 DB,  OR     .2307+000,  PH A =   -7.331 DEGREES,  GR A =   16.66 DEGREES.
     1600.829         .125         .18299-020        -.38024-010          .19598-010
     1601.029         .125         .48884-020        -.58288-010          .38613-010
     1601.229         .125         .52029-020        -.55476-010          .46101-010
     1601.429         .125         .24051-020        -.34234-010          .35116-010
     1601.629         .125         .83588-022        -.60615-011          .68444-011

SOURCE AND RECEIVERS
     100.00      500.00     1000.00     2000.00     3000.00     4000.C0     50C0.00

      .172-20      .152-14       .000+00       .000+00       .000+00       .000+00
      -.0002       -.0002      269.9995      269.9995      269.9995      269.9995

      .345-03      .356-03       .249-03       .161-03       .650-03       .183-10
       .0574        .0053        -.1149      179.5611         .0729       28.8081

      .469-03      .431-03       .453-03       .376-03       .561-03       .242-03
      -.4893     182.1377       -3.6375      -13.4081        -.1417      175.5703
        3 MODES IN SUM


     KEYS     0    0    0    0    0    0    0    0    0


     @BRKPT PRINTS
```

APPENDIX C: HANKEL FUNCTION PARAMETERS

This appendix gives the FORTRAN statements for two programs associated with the modified Hankel functions. Program PWRTRN computes the power series coefficients, d_m, from eq (57), then determines the truncation points from eq (59). The truncation points for the other three sets of coefficients can be determined by changing line 9. Different computer word lengths can be accommodated by changing line 16.

The second program, CFC, determines the asymptotic series coefficients C_m from eq (72), then determines the continued fraction coefficients as indicated by eq (81)–(83). The second set of coefficients can be determined by changing the 4 in line 11 to a 16.

```
      PROGRAM PWRTRN
C ** THIS PROGRAM DETERMINS TRUNCATION POINTS FOR THE POWER SERIES.
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION D(50), ALOGD(50)
      D(1) = 1.
      ALOGD(1) = 0.
      P = 3.
      DO 50 I = 2,50
      D(I) = D(I-1) / P / (P-2)
      P = P + 3.
      IF (D(I) .LE 0.) GO TO 50
      ALOGD(I) = ALOG10(D(I))
   50 CONTINUE
      PRINT 60, D, ALOGD
   60 FORMAT (10E12.6)
      DH = 18.
      M = 1
      DO 10 K = 2,50
   30 P = M - K
      Z = (ALOGD(K) - ALOGD(M) + DH) / 3. / P
      IF (P .GT. -1.1) GO TO 20
      A = ALOGD(M) - ALOGD(M+1) - 3. * Z
      IF (A .GT. 0.) GO TO 20
      M = M + 1
      GO TO 30
   20 L = K - 1
      MM = M - 1
      AZ = EXP (Z * 2.3025851)
      AZSQ = AZ * AZ
      PRINT 40, L, MM, Z, AZ, AZSQ
   40 FORMAT (2I5, 4E15.8)
   10 CONTINUE
      END
```

```
      PROGRAM CFC
C ** THIS PROGRAM COMPUTES A SET OF SERIES COEFFICIENTS AND THEN
C ** COMPUTES THE CORRESPONDING CONTINUED FRACTION COEFFICIENTS.
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION COEF(21,23,3), CHECK(20), C(82), S(10), A(20), B(20)
      C(1) = 1.
      BOTTOM = 1.
      TOP = 1.
      DO 2 I = 1,45
      X = 48 * I
      Y = 9 * (I + I - 1)**2 - 4
      C(I+1) = C(I) * Y / X
2     CONTINUE
      PRINT 20, (C(I), I = 1,40)
20    FORMAT (5G20.9)
11    FORMAT (/)
      DO 100 I = 1,11
      COEF(I,I,3) = 0.
      COEF(I,I+1,3) = 0.
      COEF(I,I+2,3) = 0.
100   CONTINUE
      A(1) = C(2)
      COEF(2,2,3) = 1.
      DO 140 I = 3,21
      DO 110 J = 2,I
      COEF(I,J,1) = COEF(I-1,J,3)
      COEF(I,J,2) = COEF(I-2,J,3)
      COEF(I,J,3) = COEF(I-1,J-1,3)
110   CONTINUE
      IF (I .EQ. 3) GO TO 150
      CON = 0.
      AT = 0.
      BT = 0.
      K = I - 3
      DO 120 J = 3,I
      K = K + 1
      CON = C(K) * COEF(I,J-1,3) + CON
      AT = C(K) * COEF(I,J-1,2) + AT
      BT = C(K) * COEF(I,J-1,1) + BT
120   CONTINUE
      PRINT 160, CON, AT, BT
      CHECK(I-2) = BT
      A(I-2) = -(CON + C(K+1)) / AT
150   CONTINUE
      CON = 0.
      AT = 0.
      BT = 0.
      K = I - 2
      DO 130 J = 3,I
      K = K + 1
      CON = C(K) * COEF(I,J-1,3) + CON
      AT = C(K) * COEF(I,J-1,2) + AT
      BT = C(K) * COEF(I,J-1,1) + BT
130    CONTINUE
      PRINT 160, CON, AT, BT
      PRINT 11
```

```
      B(I-2) = -(CON + A(I-2) * AT + C(K+1)) / BT
      DO 140 J = 2,I
      COEF(I,J,3) = COEF(I,J,3) + A(I-2) * COEF(I,J,2) + B(I-2) *
     * COEF(I,J,1)
140   CONTINUE
      PRINT 20, A, B, CHECK
160   FORMAT (5G20.9)
      K = -2
      J = 0
      DO 30 M = 1,18,3
      J = J + 3
      K = K + 3
      PUNCH 200, (A(I), I = K, J)
      PUNCH 200, (B(I), I = K, J)
30    CONTINUE
200   FORMAT (5X, 1H*, 3(E21.15, 1H, ))
      END
```

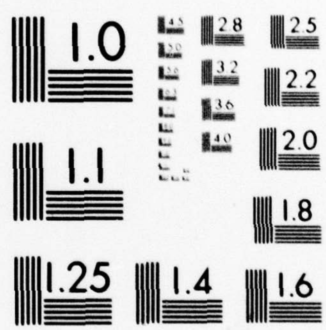
APPENDIX D: MODE FOLLOWER PROGRAM IN FORTRAN

The FORTRAN statements of the Mode Follower program are given here. This is the main body of the program. The following auxiliary subroutines from appendix A are required: SETUP, DET, HANKEL, and CFR.

AD-A072 201 NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA F/G 17/1
UNDERWATER SOUND PROPAGATION-LOSS PROGRAM. COMPUTATION BY NORMA--ETC(U)
MAY 79 D F GORDON
UNCLASSIFIED NOSC-TR-393 NL

2 OF 2
AD
A072201

END
DATE
FILMED
9-79
DDC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

```
      PROGRAM MFOLLO
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /INPUT/ Z(10),N,OMEGA,V,VI,GCU(10),GSQ(10),CAY(10),LAMBDA,L
     1AMBDI,G(10),RHO(10),GI(10),GSQI(10),CAYI(10)
      COMMON /DETMNT/ A(21,4),Q(21,4)
      REAL INCA, INCB, INCC, INCD, INCE, LAMBDA, LAMBDI
      DIMENSION T(4), PV(4), W(8), WI(8), CB(10), CBI(10), C(10),
     1  CAY SQ(10), GAMMA(10), DPK(10), GCUI(10), CI(10), CR(10),PVI(4)
     2 , CAYSQI(10), SR(4), SI(4)
      CHNG = 1. / 8192.
      CHNGI = 0.
    4 CONTINUE
C** K0 - TOTAL STEP LIMIT, K1, K2 PRINT KEYS, K3 = 1 KEEPS SAME PROFILE
C**    FOR NEXT RUN.
      READ 10, K0, K1, K2, K6, K3, TLIM, BLIM, RATIO, EX
      PRINT 10,K0, K1, K2, K6, K3, TLIM, BLIM, RATIO, EX
   10 FORMAT (5I4, 4E10.1)
      IF (TLIM .EQ. 0.) TLIM = 1.E-5
      IF (BLIM .EQ. 0.) BLIM = 1.E-2
      IF (EX .EQ. 0.) EX = 28.
      RLIM = 10.**EX
      IF (RATIO .EQ. 0.) RATIO = 2.
      IF (K0 .EQ. 0) K0 = 300
      IF (K3 .NE. 0) GO TO 128
   30 READ 1240, N,FREQ    ,ATTEN
C** STOP IF N = 0. THIS IS THE ONLY PROGRAMED STOP.
      IF (N.EQ.0) GO TO 1200
      PRINT 1250, N,FREQ
C** PARAMETERS READ IN BELOW ARE THOSE AT THE TOP OF EACH LAYER.
C** READ IN VELOCITIES.
      READ 1260, (C(I),I=1,N)
      PRINT 1280, (C(I),I=1,N)
C** READ IN DEPTHS.
      READ 1260, (Z(I),I=1,N)
      PRINT 1280, (Z(I),I=1,N)
C** READ IN GRADIENTS
      READ 1260, (GAMMA(I),I=1,N)
      PRINT 1280, (GAMMA(I),I=1,N)
C** READ IN ATTENUATION FACTOR IN LOSS PER KILOMETER.
      READ 1260, (DPK(I),I=1,N)
      PRINT 1280, (DPK(I),I=1,N)
C** READ IN DENSITIES (BLANK INPUT IMPLIES SEA WATER DENSITY).
      READ 1260, (RHO(I),I=1,N)
      PRINT 1280, (RHO(I),I=1,N)
  128 CONTINUE
      NUMBER = 1
      JX = 0
C**  NX = VARIABLE, NY = LAYER NUMBER, NZ = CONTINUITY
      READ 119, NX, NY, NZ, PK, VALL,DP, V, VI, STEP, STEPI
      PRINT 21, NX, NY, NZ, PK, VALL,DP, V, VI, STEP, STEPI
  119 FORMAT (3I2, 4X, 7D10.2)
   21 FORMAT (10H  VARIABLE ,I2, 10H  LAYER NO , I2,12H  CONTINUITY
     * I2, / 7G15.5)
      PK = PK - DP
C** START NEW CYCLE BY INCREMENTING VARIABLE.
  107 PK = PK + DP
```

```
      IF (DP) 108,999,109
C** CHECK IF DESIRED LIMIT OF VARIABLE HAS BEEN REACHED.
 108  IF (PK .LT. VALL) GO TO 3
      GO TO 133
 109  IF (PK .GT. VALL) GO TO 3
 133  GO TO (131,101,102,103,104,105),NX
 131  FREQ = PK
      GO TO 106
 101  C(NY) = PK
      IF (NZ .NE. 0) GO TO 106
 134  IF (NY .EQ. N) GO TO 135
      GAMMA(NY) = 0.
      IF (NY .LT. 2) GO TO 106
 135  GAMMA(NY-1) = 0.
      GO TO 106
 102  Z(NY) = PK
      IF (NZ .EQ. 1) GO TO 106
      IF (NY .LT. N) GO TO 134
      IF (NUMBER .EQ. 1) GO TO 106
      C(NY) = 0.
      GO TO 106
 103  GAMMA(NY) = PK
      IF (NZ .NE.0) GO TO 106
      J = NY + 1
      DO 121 I = J,N
      C(I) = 0.
 121  CONTINUE
 104  DPK(NY) = PK
      GO TO 106
 105  RHO(NY) = PK
 106  CONTINUE

C** COMPLETE PROFILE **
      DO 100 I=1,N
C** SET UNSPECIFIED DENSITIES TO 1.02 (SEA WATER).
      IF (RHO(I).NE.0.) GO TO 40
      RHO(I)=1.02
40    IF (I.EQ.1) GO TO 50
C**     COMPUTE VELOCITY AT BOTTOM OF PREVIOUS LAYER.
      TEMP=CI(I-1)**2
      TEMDR=C(I-1)**2
      TEMDI=(TEMDR+TEMDR+TEMDR-TEMP)*CI(I-1)
      TEMDR=(TEMDR-TEMP-TEMP-TEMP)*C(I-1)
      TEMP=(GAMMA(I-1)+GAMMA(I-1))*(Z(I)-Z(I-1))-C(I-1)
      TEMDEN=TEMP**2+CI(I-1)**2
      TEM1=(TEMDI*CI(I-1)-TEMDR*TEMP)/TEMDEN
      TEM1I=(-TEMDI*TEMP-TEMDR*CI(I-1))/TEMDEN
     CB(I)=SQRT(.5*(TEM1+SQRT(TEM1**2+TEM1I**2)))
      CBI(I)=TEM1I/(CB(I)+CB(I))
50    IF (C(I).NE.0) GO TO 60
C** IF VELOCITY WAS UNSPECIFIED USE VELOCITY AT BOTTOM OF PREVIOUS LAYER
      C(I)=CB(I)
60    IF (DPK(I).NE.0.) GO TO 70
      CI(I)=0.
      GO TO 80
C** IF ATTENUATION IS TO BE APPLIED TO A LAYER, COMPUTE COMPLEX VELOCITY
C** KEEP ABSOLUTE C EQUAL TO GIVEN REAL C FOR SIMPLICITY.
```

```
70    TEMP=54578.*FREQ
        TEMDI=DPK(I)*C(I)
        TEMDR=TEMP**2+TEMDI**2
        CI(I)=TEMDI*TEMP*C(I)/TEMDR
        C(I)=TEMP**2*C(I)/TEMDR
80      IF (GAMMA(I).NE.0.) GO TO 100
        IF (I.EQ.N) GO TO 90
C** COMPUTE GRADIENT IF NOT GIVEN.
        GAMMA(I)=(C(I+1)**2-C(I)**2)*C(I)/(2.*C(I+1)**2*(Z(I+1)-Z(I)))
        IF (I.EQ.N) GO TO 90
        GO TO 100
C** REDUCE LAYERS BY ONE IF FINAL POINT ONLY DEFINES GRADIENT IN LAST LAYER.
90      N=N-1
100     CONTINUE

C** COMPUTE USEFULL QUANTIES **
      OMEGA=6.283185307*FREQ
      DO 120 I=1,N
        TEMP=C(I)**2+CI(I)**2
        CAY(I)=OMEGA*C(I)/TEMP
        CAYI(I)=-OMEGA*CI(I)/TEMP
        CAYSQ(I)=CAY(I)**2-CAYI(I)**2
      CAYSQI(I)=2.*CAY(I)*CAYI(I)
        TEMDR=-2.*GAMMA(I)*CAYSQ(I)
      TEMDI=-2.*GAMMA(I)*CAYSQI(I)
        GCU(I)=(TEMDR*C(I)+TEMDI*CI(I))/TEMP
        GCUI(I)=(TEMDI*C(I)-TEMDR*CI(I))/TEMP
      TEMP=EXP(ALOG(GCU(I)**2+GCUI(I)**2)/6.)
      GI(I)=TEMP*SIN(ATAN2(GCUI(I),ABS(GCU(I)))/3.)
      G(I)=SQRT(TEMP**2-GI(I)**2)
        IF (GAMMA(I).LT.0.) GO TO 110
        G(I)=-G(I)
110     GI(I)=-GI(I)
C** XM IS A LAYER STRENGH PARAMETER USED ONLY TO COMPARE WITH OTHER MODE
        XMI=-GI(I)*(Z(I+1)-Z(I))
        XM=-G(I)*(Z(I+1)-Z(I))
        GSQI(I)=2.*G(I)*GI(I)
120     GSQ(I)=G(I)**2-GI(I)**2
      IF (JX .GT. 0) GO TO 113
C** GO TO INITIAL 3 STEPS OR TO THE STANDARD STEP.
      IF (NUMBER - 4) 71,111,122
 71   CALL SETUP
      CALL DETNT(N,DET,DETI)
        VEL=DET
        VELI=DETI
        DELTA=STEP
        DELTI=STEPI
        IF(DELTA.NE.0.)GO TO 250
        IF(DELTI.EQ.0.)DELTA=.01
250     SIZE2=100.
        IF (K6.LT.3) PRINT 1320, V,VI,DET,DETI,A(21,4),Q(21,4)
C** ITERATE FOR MODE UP TO 7 STEPS.
        DO 310 J=1,12
          V=V+DELTA
          VI=VI+DELTI
C** DO NOT PERMIT IMAGINARY PART TO BECOME NEGATIVE.
          IF (VI) 260,270,280
```

```
260       DELTI=DELTI-VI
270   VI=1.E-18
C** SET UP DETERMINANT FOR PHASE VELOCITY V + VI
280       CALL SETUP
          CALL DETNT (N,DET,DETI)
          IF (K6.NE.1) GO TO 300
      PRINT 1330, V, VI, DET, DETI, SLR, SLI
 300  TEMNR = DELTA
      TEMNI = DELTI
          TEMDR=VEL-DET
          TEMDI=VELI-DETI
           TEMDEN=TEMDR*TEMDR+TEMDI*TEMDI
          IF (TEMDEN.EQ.0.) GO TO 320
          TEMRNU=TEMNR*TEMDR+TEMNI*TEMDI
          TEMINU=TEMNI*TEMDR-TEMNR*TEMDI
      SLR = TEMRNU / TEMDEN
      SLI = TEMINU / TEMDEN
      IF (J .GT. 3) GO TO 125
      SR(4-NUMBER) = SLR
      SI(4-NUMBER) = SLI
 125  DELTA = DET * SLR - DETI * SLI
      DELTI = DET * SLI + DETI * SLR
          SIZE=DELTA*DELTA+DELTI*DELTI
C** DISCONTINUE ITERATION AFTER 2ND STEP IF CORRECTION STEP IS MORE THAN
C** PREVIOUS STEP.
          IF ((SIZE.GT.SIZE2).AND.(J.GT.2)) GO TO 320
          SIZE2=SIZE*2.
          VEL=DET
          VELI=DETI
310       CONTINUE
 320   CONTINUE
 51   PV(4-NUMBER) = V
      PVI(4-NUMBER)= VI
      NUMBER = NUMBER + 1
      GO TO 107
C** START STANDARD STEP, EXTRAPOLATE PHASE VELOCITY AND SLOPE.
 111  INCA = DP
      INCB = DP
      INCC = DP
 122  INCD = -INCB - INCC
      T(1) = INCB * INCD
      T(2) = INCB * INCC
      T(3) = INCD * INCC
      DO 112 IS = 1,3
      W(IS+4) = -SR(IS) / T(IS)
      WI(IS + 4) = -SI(IS) / T(IS)
      W(IS) = -PV(IS) / T(IS)
 112  WI(IS) = -PVI(IS) / T(IS)
 113  INCD = INCA + INCB
      INCE  = INCD + INCC
      T(1) = INCD * INCE
      T(2) = INCA * INCE
      T(3) = INCA * INCD
      SLOP = 0.
      SLOPI = 0.
      SUM = 0.
      SUMI = 0.
```

```
      DO 114 IS = 1,3
      SLOP = SLOP + W(IS + 4) * T(IS)
      SLOPI = SLOPI + WI(IS+4) * T(IS)
      SUM = SUM + W(IS) * T(IS)
  114 SUMI = SUMI + WI(IS) * T(IS)
      V = SUM
      VI = SUMI
      CALL SETUP
      CALL DETNT (N,DET,DETI)
C** EVALUATE DETERMINANT AT THE EXTRAPOLATED POINT.
      VEL = DET
      VELI = DETI
C** ITERATE FOR THE ROOT USING EXTRAPOLATED SLOPE.
      DELTA = DET * SLOP - DETI * SLOPI
      DELTI = DET * SLOPI + DETI * SLOP
      IF (K1 .EQ. 1) PRINT 1330, V, VI, DET, DETI, DELTA, DELTI
      V = V + DELTA
      VI = VI + DELTI
      IF (VI .GE. 0.) GO TO 124
      DELTI = DELTI - VI
      CHNGI = CHNGI - VI
      VI = 0.
C** RE-EVALUATE AT NEW POINT.
  124 CALL SETUP
      CALL DETNT (N, DET, DETI)
      TEMNR = DELTA
      TEMNI = DELTI
          TEMDR=VEL-DET
          TEMDI=VELI-DETI
           TEMDEN=TEMDR*TEMDR+TEMDI*TEMDI
      IF (TEMDEN .EQ. 0.) GO TO 123
          TEMRNU=TEMNR*TEMDR+TEMNI*TEMDI
          TEMINU=TEMNI*TEMDR-TEMNR*TEMDI
C** EVALUATE SLOPE (RECIPROCAL ACTUALLY USED).
      SLR = TEMRNU / TEMDEN
      SLI = TEMINU / TEMDEN
      DELTA = DET * SLR - DETI * SLI
      DELTI = DET * SLI + DETI * SLR
      IF (K1 .EQ. 1) PRINT 1330, V, VI, DET, DETI, DELTA, DELTI
C** CORRECT PHASE VELOCITY TO BEST VALUE.
      V = V + DELTA
      VI = VI + DELTI
      TEMP = V**2 / (TEMNR**2 + TEMNI**2)
C** WAS INCREMENT LARGE ENOUGH TO PERMIT EVALUATION OF SLOPE.
      IF (TEMP .LT. RLIM) GO TO 123
      IF (TEMP .LT. 1.E34) GO TO 141
      SLR = SLOP
C** IF NOT, USE EXTRAPOLATED SLOPE.
      SLI = SLOPI
      GO TO 141
  123 CONTINUE
C** IF SO, FIND 1 - RATIO OF SLOPES.
      TEMDEN = (SLR**2 + SLI**2)
      TEMDR = SLR * SLOP + SLI * SLOPI - TEMDEN
      TEMDI = SLR * SLOPI - SLI * SLOP
      TEMP = (TEMDR**2 + TEMDI**2) / TEMDEN**2
      IF (TEMP .GT. TLIM) GO TO 116
```

```
C** SLOPE RATIO TOO GOOD.  DOUBLE STEP.
 141  DP = DP * RATIO
      GO TO 117
 116  IF (TEMP .LT. BLIM) GO TO 117
      PRINT 130, PK,V,VI,DET,DETI,SLR,SLI,TEMP,DBLOSS,NUMBER
 130  FORMAT (1X,E14.6,E16.9,E13.7,6E10.3,I5)
C** SLOPE RATIO TOO POOR.  HALVE STEP.
      IF (NUMBER .LT. 7) GO TO 126
      PK = PK - DP
      DP = DP / RATIO
      INCA = DP
      JX = JX + 1
      IF (K2 .EQ. 1) PRINT 118, PK, V, VI, DET, DETI
C** STOP ON 5 SUCCESSIVE FAILURES.  MODE IS LOST.
      IF (JX .LT. 5) GO TO 107
      PRINT 810, N, FREQ
 810  FORMAT (I4, G12.5)
 3    DO 801 I = 1,N
      PRINT 800, C(I), Z(I), GAMMA(I), DPK(I), RHO(I), G(I)
 800  FORMAT (10G12.5)
 801  CONTINUE
      GO TO 4
 126  PRINT 127 , N, TEMP
 127  FORMAT (7H NUMBER, I3, 22H FAILED, SLOPE RATIO  ,F10.6)
C** UPDATE ALL QUANTITIES FOR NEXT STEP.
 117  INCC = INCB
      INCB = INCA
      INCA = DP
      PV(3) = PV(2)
      PVI(3) = PVI(2)
      PV(2) = PV(1)
      PVI(2) = PVI(1)
      PV(1) = V
      PVI(1) = VI
      JX = 0
      DENOM = V * V + VI * VI
      LAMBDI = -OMEGA * VI / DENOM
      DB LOSS = -8686. * LAMBDI
      SR(3) = SR(2)
      SR(2) = SR(1)
      SR(1) = SLR
      SI(3) = SI(2)
      SI(2) = SI(1)
      SI(1) = SLI
      GV = V**2 / (V - FREQ * (V-PV(2)) / INCB)
      PRINT 118, PK,V,VI,DET,DETI,SLR,SLI,TEMP,DBLOSS,GV,NUMBER
 118  FORMAT (E15.7,E16.9,E13.7,6E10.3,F11.5,I5)
      NUMBER = NUMBER + 1
C** CHECK TOTAL NUMBER OF STEPS.
      IF (NUMBER .GT. K0) GO TO 3
      GO TO 107
 999  STOP
1250  FORMAT (I3,8H LAYERS,,F10.1,3H HZ)
1260  FORMAT(8E10.4)
1270  FORMAT (8H ATTEN =,G10.5,5HDB/KM)
1280  FORMAT (8F14.5)
1320      FORMAT (/,6E18.9)
```

```
      1330  FORMAT (6E18.9)
      1240  FORMAT(I2,E10.1, E10.2)
      1290  FORMAT (7X,6H RE M ,8X,6H IM M ,8X,6H L/KYD,8X,6H RE C ,8X,6H IM C
           1 ,5X,12H RE C BOTTOM,4X,12H IM C BOTTOM)
       1200 CONTINUE
            END
```

INITIAL DISTRIBUTION

ASST. SECRETARY OF THE NAVY (R&ES)

OFFICE OF NAVAL RESEARCH
- ONR-102-OS
- ONR-480
- ONR-485

CHIEF OF NAVAL OPERATIONS
- NOP-095
- NOP-951E

CHIEF OF NAVAL MATERIAL
- NMAT-08T2
- ASW-122

NAVAL RESEARCH LABORATORY
- CODE 8100
- CODE 8120

NAVAL OCEAN RESEARCH & DEVELOPMENT ACTIVITY
- CODE 320
- CODE 460
- CODE 500
- CODE 520
- CODE 600
- CODE 200

NAVAL OCEANOGRAPHY COMMAND

NAVAL OCEANOGRAPHIC OFFICE
- CODE 3440

NAVAL ELECTRONIC SYSTEMS COMMAND
- PME-124
- NELEX-320

NAVAL SEA SYSTEMS COMMAND
- NSEA-003
- NSEA-63R
- NSEA-63R-23
- NSEA-63D
- PMS-407

NAVAL AIR SYSTEMS COMMAND
- NAIR-370

NAVAL AIR DEVELOPMENT CENTER

NAVAL COASTAL SYSTEMS CENTER

NAVAL INTELLIGENCE SUPPORT CENTER

NAVAL POSTGRADUATE SCHOOL

NEW LONDON LABORATORY
- NAVAL UNDER WATER SYSTEMS CENTER

NAVAL UNDERWATER SYSTEMS CENTER

WHITE OAK LABORATORY
- NAVAL SURFACE WEAPONS CENTER

D. W. TAYLOR NAVAL SHIP RESEARCH
AND DEVELOPMENT CENTER

FLEET NUMERICAL WEATHER CENTRAL

WOODS HOLE OCEANOGRAPHY INSTITUTION
WOODS HOLE, MA 02543

MARINE PHYSICAL LABORATORY
UNIVERSITY OF CALIFORNIA, SAN DIEGO
BLDG. 106
SAN DIEGO, CA 92152

APPLIED RESEARCH LABORATORY
THE PENNSYLVANIA STATE UNIVERSITY
P.O. BOX 30
STATE COLLEGE, PA 16801

THE UNIVERSITY OF TEXAS
APPLIED RESEARCH LABORATORY
P.O. BOX 8029
AUSTIN, TX 78712

APPLIED PHYSICS LABORATORY
UNIVERSITY OF WASHINGTON
1013 N.E. 40TH ST.
SEATTLE, WA 98195

NATIONAL RESEARCH COUNCIL
ATTN: COMMITTEE UNDERSEA WARFARE
2101 CONSTITUTION AVE., NW
WASHINGTON, DC 20418

DEFENSE DOCUMENTATION CENTER (12)